LOGIC LAB

Manuale utente

Rev. 4.0 - Marzo 2012

SOLUZIONI SOFTWARE PER L'AUTOMAZIONE INDUSTRIALE

Contents

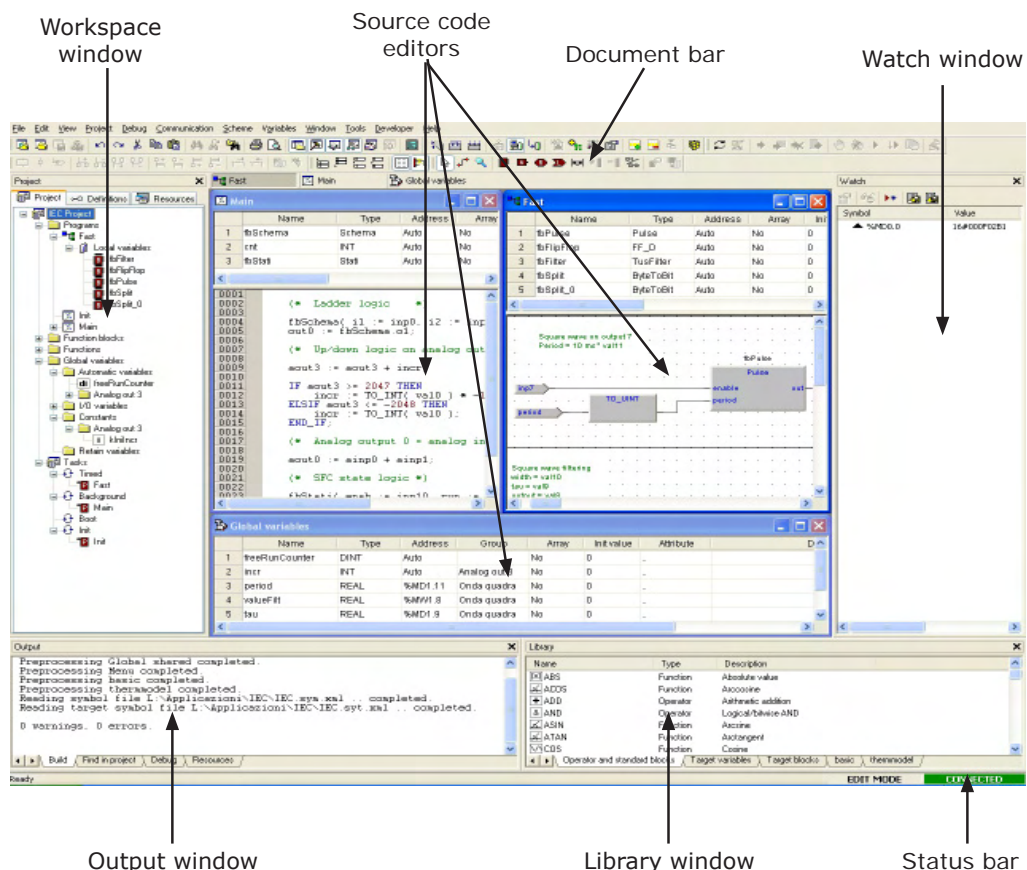
1. VISIONE DI INSIEME

LogicLab è un IEC61131-3 Integrated Development Environment che supporta l'intera serie di linguaggi definiti dallo standard.

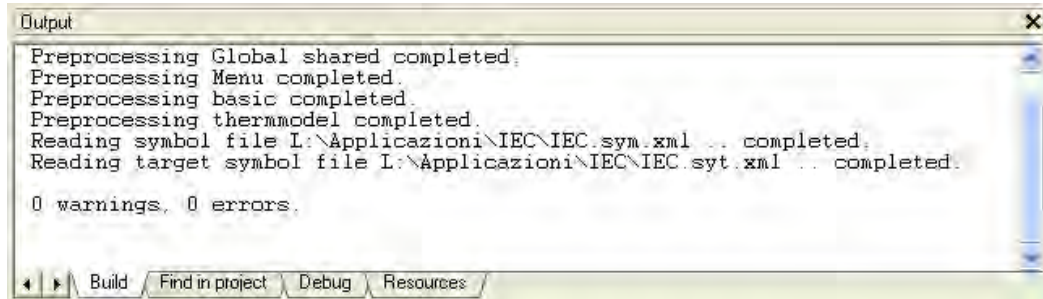
Per il supporto dell'utente in tutte le attività coinvolte nello sviluppo di un'applicazione, LogicLab include:

- editor del codice sorgente testuale per i linguaggi di programmazione Instruction List (abbreviato, IL) e Structured Text (abbreviato, ST) (vedi cap.6) ;
- editor del codice sorgente grafico per i linguaggi di programmazione Ladder Diagram (abbreviato, LD), Function Block Diagram (abbreviato, FBD), e il Sequential Function-Chart (abbreviato, SFC) (vedi cap.6); editor del codice sorgente grafico per i linguaggi di programmazione Ladder Diagram (abbreviato, LD), Function Block Diagram (abbreviato, FBD), e il Sequential FunctionChart (abbreviato, SFC) (vedi cap.6);
- un compilatore, che traduce direttamente in codice macchina le applicazioni scritte secondo lo standard IEC, evitando il bisogno di un traduttore run-time, permettendo l'esecuzione del programma il più velocemente possibile (vedi cap.7);
- un sistema di comunicazione che permette il download dell'applicazione al target environment (vedi cap.8);
- un ricco set di strumenti di debug , che variano da una finestra watch facilmente utilizzabile a strumenti più potenti, che permettono di campionare i dati che cambiano velocemente direttamente sul target environment, assicurando un'informazione accurata ed affidabile (vedi cap.9)

1.1 AREA DI LAVORO

La figura sottostante mostra un esempio dello spazio di lavoro LogicLab, includendo molte delle sue componenti più usate.

I paragrafi seguenti mostrano una panoramica di questi elementi.

1.1.1 LA FINESTRA DI OUTPUT

La finestra di *Output* è il luogo in cui LogicLab mostra i suoi messaggi di output. Questa finestra contiene quattro schede: *Build*, *Find in project*, *Debug* e *Resources*.

Build

Il pannello *Build* mostra l'output delle seguenti attività:

- apertura di un progetto;
- compilazione di un progetto;
- download codice ad un target.

Find in project

Questo pannello mostra i risultati della ricerca con *Find in project*.

Debug

Il pannello *Debug* mostra le informazioni sulle attività di debug avanzate (per esempio, i breakpoint).

Resources

Il pannello *Resources* mostra i messaggi legati al target specifico con cui LogicLab si sta connettendo.

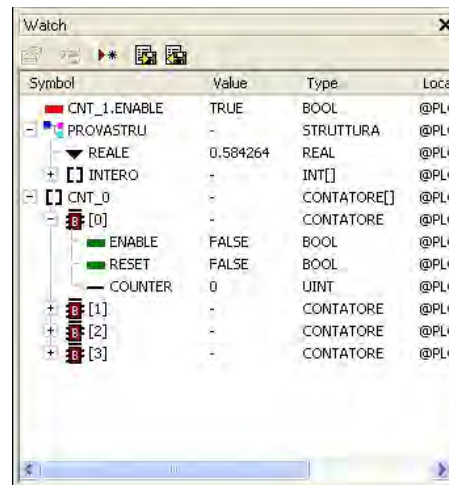
1.1.2 LA BARRA DI STATO

La barra *Status* mostra nella parte sinistra lo stato dell'applicazione, e un comando animato che riporta lo stato della comunicazione nella parte destra.

1.1.3 LA BARRA DEI DOCUMENTI

La barra *Document* elenca l'insieme dei documenti aperti al momento per la modifica in LogicLab.

1.1.4 LA FINESTRA WATCH

La finestra `Watch` è uno dei numerosi strumenti di debug forniti da LogicLab. Tra gli altri strumenti di debug è importante menzionare l'oscilloscopio (vedi paragrafo 9.2), i trigger e la modalità di debug live (vedi paragrafo 9.4).

1.1.5 LA FINESTRA DELLE LIBRERIE

La finestra `Library` contiene una serie di pannelli diversi, che rientrano nelle categorie spiegate nei paragrafi seguenti.

E' possibile scegliere la modalità di visualizzazione cliccando il tasto destro del mouse. Nella modalità `View list`, ciascun elemento è rappresentato dal nome e dall'icona. Invece, nella modalità `View details` appare una tabella, ogni riga della quale è associata ad uno degli elementi inseriti. Quest'ultima modalità mostra inoltre il `Type` (Operatore/Funzione) e la descrizione di ciascun elemento.

Cliccando col tasto destro uno degli elementi di questo pannello, e cliccando successivamente `Object Properties` nella finestra di dialogo, appare una finestra contenente ulteriori informazioni sugli elementi selezionati (tipi di input e output supportati, nome dei pins di input e output, ecc.).

1.1.5.1 OPERATORS AND STANDARD BLOCKS

Questo pannello elenca gli elementi del linguaggio base, come gli operatori e le funzioni definiti dallo standard IEC 61131-3.

1.1.5.2 TARGET VARIABLES

Questo pannello elenca tutte le variabili di sistema, chiamate anche variabili del target, che sono l'interfaccia tra il firmware e il codice di applicazione PLC.

Name	Type	Address	Group	Description
Ad_InPo	INT	%MW0.1	DEB - ANALOG-DIGITAL EN...	incremental position
Ad_NuCi	INT	%MW0.12	DEB - ANALOG-DIGITAL EN...	DSP cycles without position increment
Ad_PeSp	DINT	%MW0.10	DEB - ANALOG-DIGITAL EN...	calculated speed
Ad_SeOf	INT	%MW0.9	DEB - ANALOG-DIGITAL EN...	sine channel offset
Ad_ViPo	DINT	%MW0.2	DEB - ANALOG-DIGITAL EN...	virtual position
Ad_ViPoIni	DINT	%MW0.218	DEB - ANALOG-DIGITAL EN...	

Operator and standard blocks | Target variables | basic | thermmodel

1.1.5.3 TARGET BLOCKS

Questo pannello elenca tutte le funzioni di sistema e i blocchi funzione disponibili sul target device specifico.

Library

Name	Type	Description
sysMsgInterpMono	Function	Checks messages from single axis int...
sysQuiesInterpMonoPlc	Function	Verifies the end of an interpolated sin...
sysResetInterpMonoPlc	Function	Resets an interpolated single axis mo...
sysSleep	Function	Puts the current task in the sleeping s...
sysStartInterpMono	Function	Starts an interpolated single axis mov...
sysStartInterpMonoPlc	Function	Starts an interpolated single axis mov...
sysTargetInterpMonoPlc	Function	Verifies if the target of an interpolated ...
sysWaitInterpMono	Function	Waits until the end of an interpolated ...

Operator and standard blocks | Target variables | Target blocks | basic

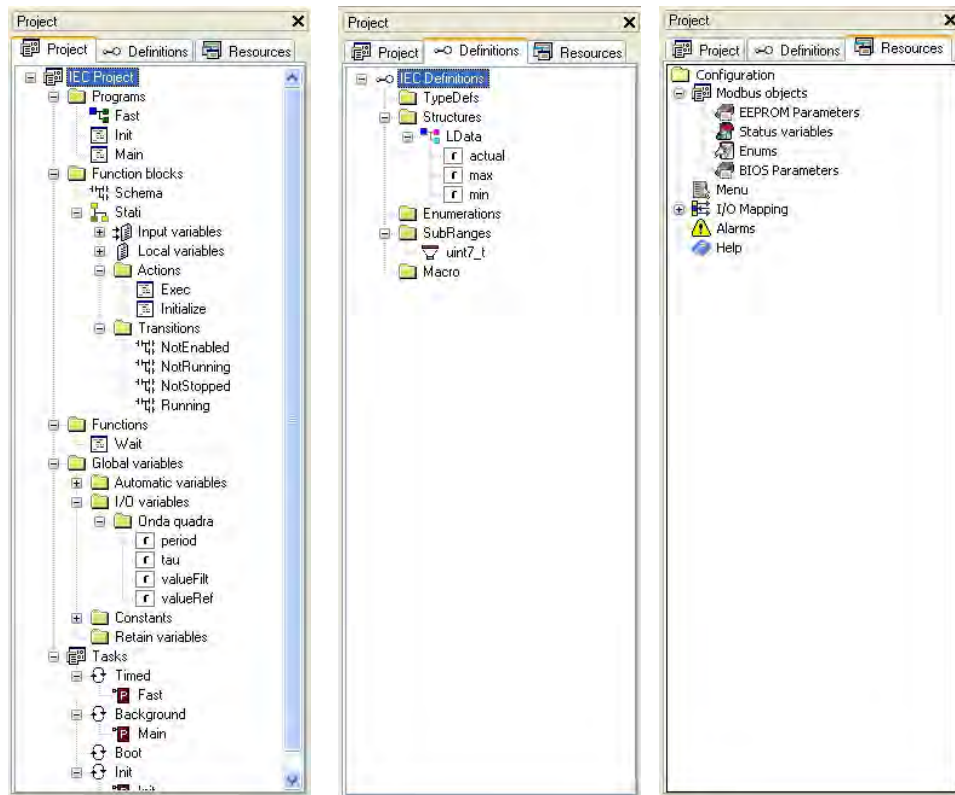
1.1.5.4 PANNELLI INCLUSI NELLA LIBRERIA

I pannelli descritti nei paragrafi precedenti sono solitamente sempre disponibili nella finestra `Library`. Tuttavia, altri pannelli possono essere aggiunti a questa finestra, uno per ogni libreria inclusa nel progetto LogicLab in corso. Per esempio, l'immagine sotto mostra un progetto LogicLab che include due librerie, `basic.pll` e `thermmodel .pll` (vedi anche paragrafo 4.7).

Name	Type	Description
BitToByte	Function	Compose a byte from 8 bits
BitToWord	Function	Compose a word from 16 bits
ByteToBit	Function block	Split a byte into bits
ByteToWord	Function	Compose a word from 2 bytes
F_TRIG	Function block	Falling edge detector
FF_D	Function block	D-type flip-flop

Operator and standard blocks | Target variables | basic | thermmodel

1.1.6 LA FINESTRA DI LAVORO

La finestra *Workspace* è formata da tre pannelli distinti, come mostrato nella figura seguente.

1.1.6.1 PROGETTO

Il pannello *Project* contiene una serie di cartelle:

- P*rogram, Function blocks, Functions:* ciascuna cartella contiene le Program Oraganization Units (abbreviato in POUs- vedi paragrafo 5.1) del tipo specificato nel nome della cartella.
- *Global Variables*: è ulteriormente suddiviso in *Variables*, *I/O Variables*, *Constants* e *Retain Variables*. Ogni cartella contiene le variabili globali del tipo specificato nel nome della cartella (vedi paragrafo 5.2).
- *Tasks*: questa voce elenca i task del sistema e i programmi associati ad ogni task (vedi paragrafo 5.3).

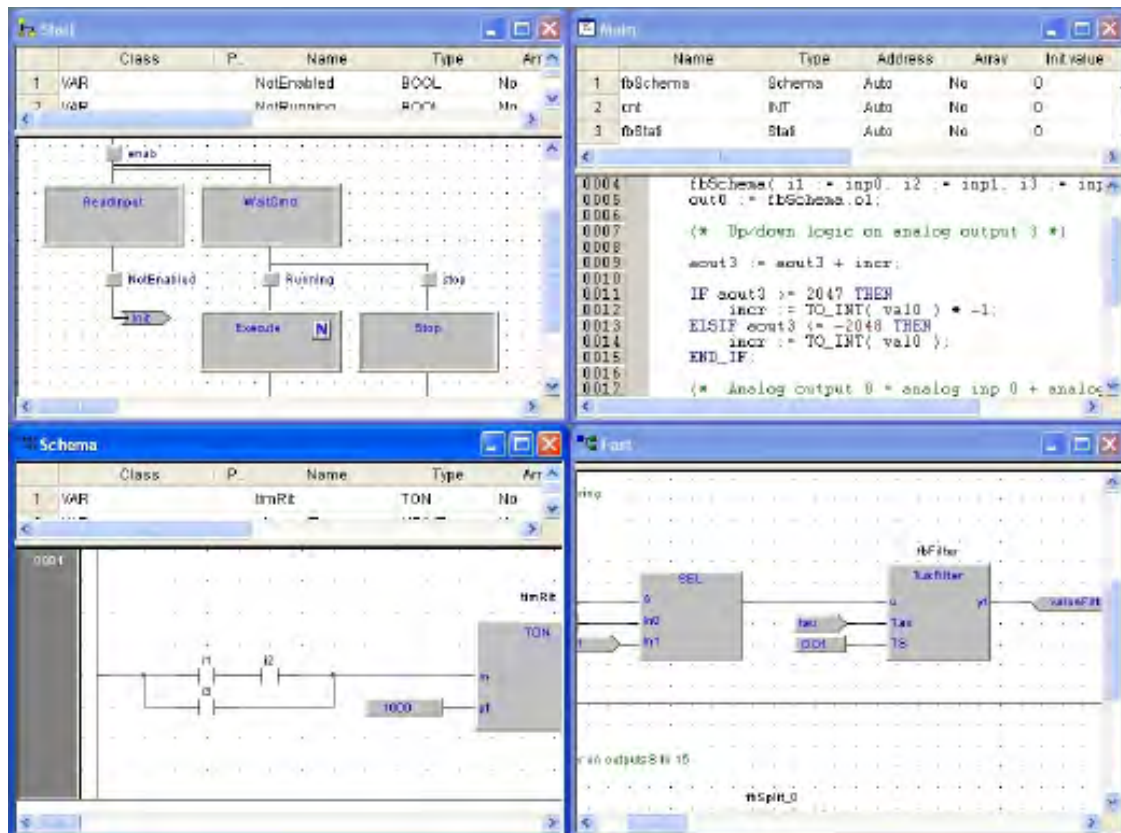
1.1.6.2 DEFINIZIONI

Il pannello *Definitions* contiene le definizioni di tutti i tipi di dati definiti dall'utente, come le strutture o i tipi enumerativi.

1.1.6.3 RISORSE

I contenuti del pannello *Resources* dipendono dal target device con cui LogicLab si sta relazionando: può includere elementi di configurazione, schemi, wizard, ecc.

1.1.7 EDITOR DEL CODICE SORGENTE

L'ambiente di programmazione LogicLab include un set di editor per gestire, modificare e stampare i file sorgente scritti in uno dei cinque linguaggi di programmazione definiti dallo standard IEC 61131-3 (vedi cap.6).

La definizione delle variabili locali e globali è supportata da specifici editor simili ad un foglio elettronico.

	Name	Type	Address	Group	Array	Init value	Attribute	Description
1	freeRunCounter	DINT	Auto		No	0		
2	incr	INT	Auto	Analog out 3	No	0	...	
3	period	REAL	%MD1.11	Onda quadra	No	0	...	
4	valueFilt	REAL	%MW1.8	Onda quadra	No	0	...	
5	tau	REAL	%MD1.9	Onda quadra	No	0	...	
6	valueRef	REAL	%MD1.10	Onda quadra	No	0	...	
7	kIniIncr	INT	Auto	Analog out 3	No	50	CONSTANT	

2. QUICK START

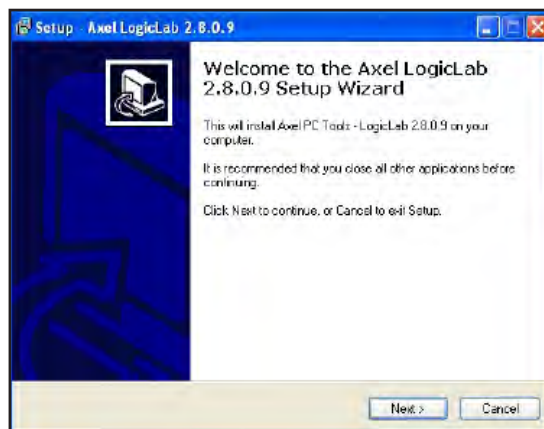
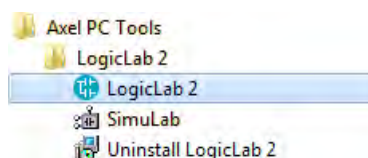
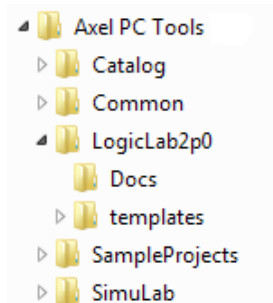
Questo capitolo è una guida che vi introdurrà passo per passo all'uso dell'ambiente di sviluppo LogicLab.

Sarete guidati attraverso tutti i compiti che dovrete assolvere per gestire e monitorare una semplice applicazione PLC.

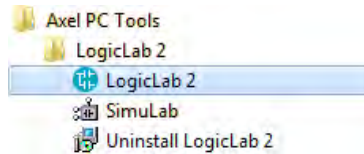
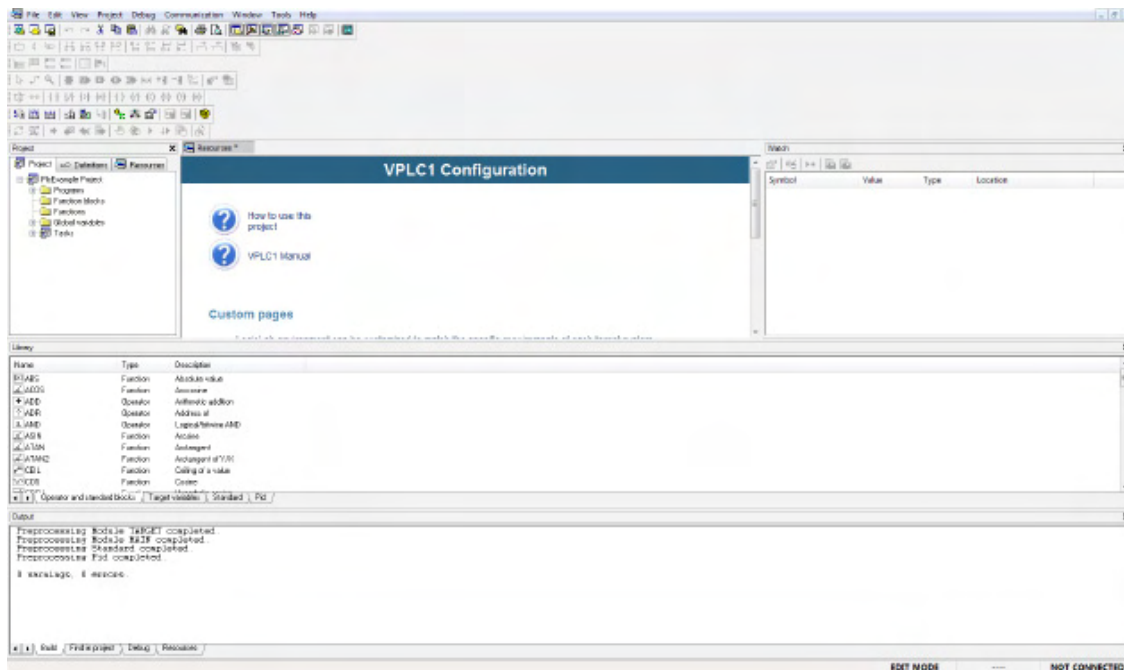
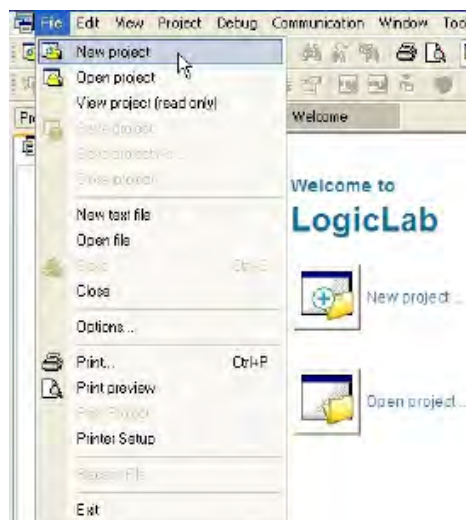
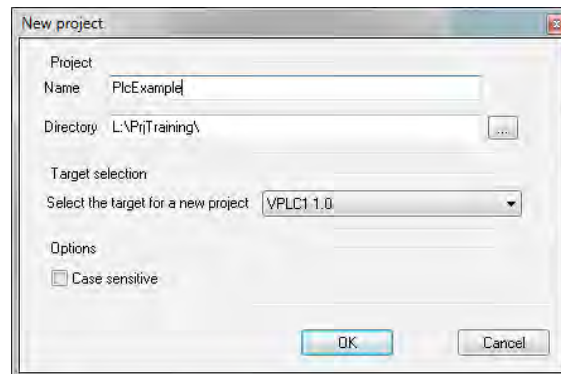
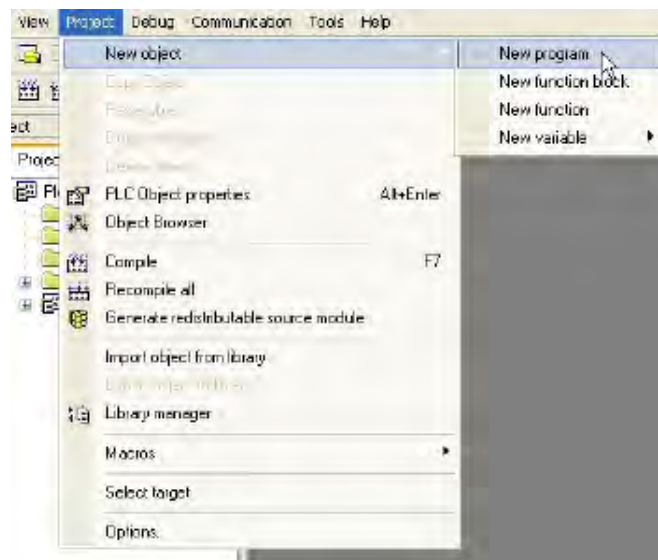
Gli esempi di questo capitolo si riferiscono ad un target virtuale, *VPLC1*, che è installato insieme a LogicLab durante la procedura di setup.

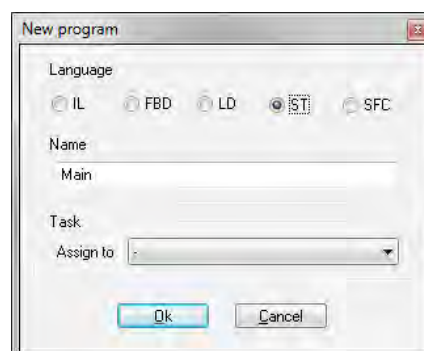
2.1 SETUP PROCEDURA

Per installare LogicLab, seguire la semplice procedura seguente:

1) Chiudere tutte le applicazioni in funzione sul sistema.
2) Aprire il file di esecuzione di setup di LogicLab e seguire le istruzione indicate nel setup wizard.

La procedura di setup aggiunge il seguente menu *Start*.

La cartella di LogicLab nel file system ha la seguente struttura.

2.2 USARE L'IDE

Le seguenti istruzioni mostrano come creare, gestire ed eseguire il debug di un semplice programma PLC di LogicLab. Il programma è scritto in linguaggio ST ed eseguito direttamente sulla vostra macchina. Un target virtuale, *VPLC1*, è stato installato durante la procedura di setup. *VPLC1* emula un controller programmabile semplice fornito del digitale e dell' analogico I/O.

1) Lanciare LogicLab dal menu *Start*.

2) Quando LogicLab è lanciato per la prima volta, non vengono mostrate tutte le finestre e le barre d'applicazione. E' possibile personalizzare lo spazio di lavoro di LogicLab attraverso il menu *View*.

3) Per creare un nuovo progetto, selezionare *New project* dal menu *File*.

4) Inserire il nome del nuovo progetto (per esempio, `PlcExample`) e selezionare la cartella di destinazione, poi premere `OK`. Per questo progetto è stato selezionato il target virtuale `VPLC1`. Lo spazio di lavoro di LogicLab ha ora l'aspetto mostrato nella figura del passaggio 2 (ovvero sono mostrati tutti gli operatori e le variabili di target, il logo dello status di precompilazione è apparso nella finestra di output).

5) Per aggiungere un nuovo programma vuoto al progetto, selezionare `New program` dal menu `Project > New object`.

Scegliere il linguaggio ST, inserire il nome del nuovo programma (per esempio, `Main`) e premere `OK`.

E' inoltre possibile assegnare qui il programma ad uno dei task disponibili.

LogicLab mostra ora la finestra di modifica dell'oggetto `Main program`. In alto c'è l'editor delle variabili locali. L'editor delle variabili permette di aggiungere, rimuovere, copiare e incollare le definizioni delle variabili.

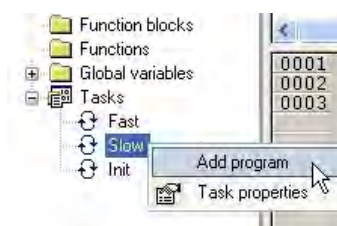
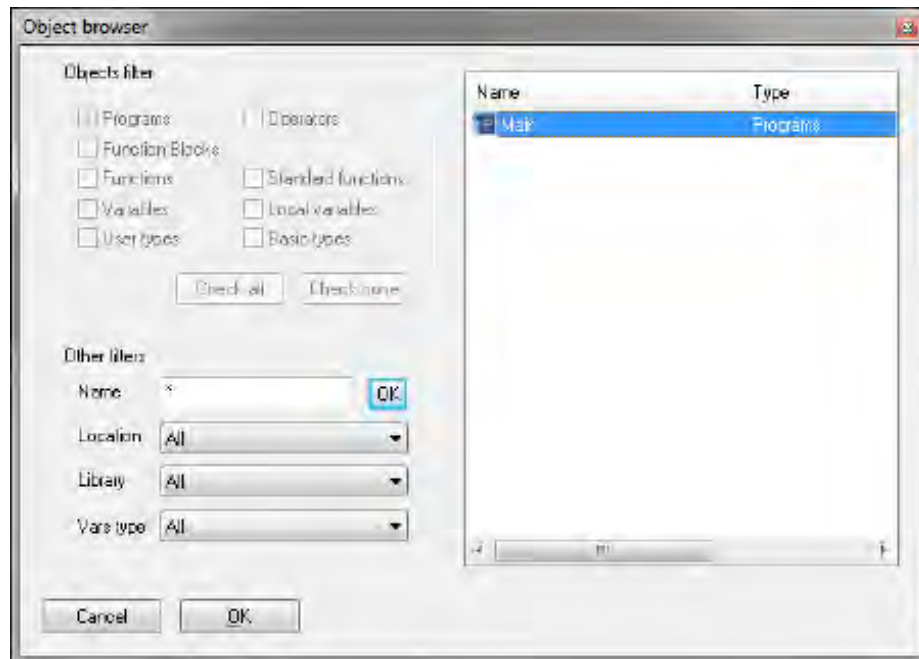
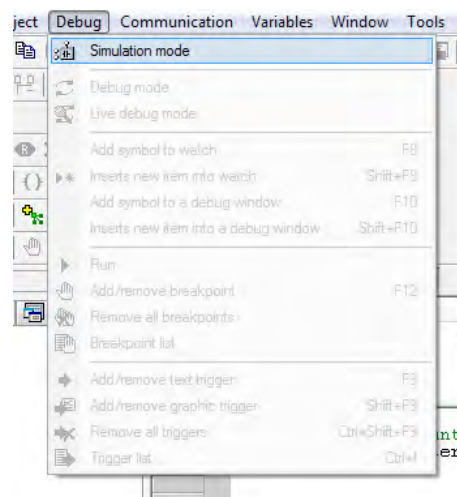
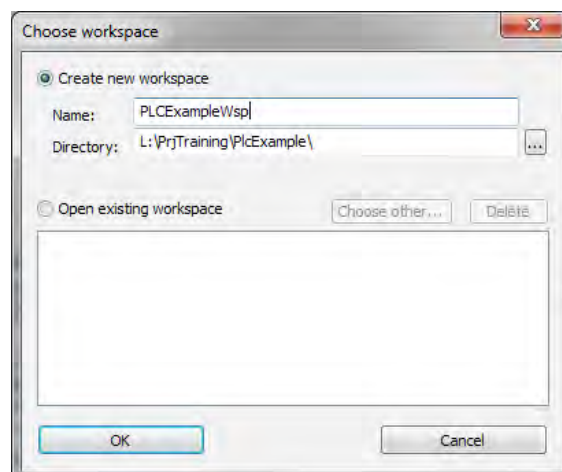
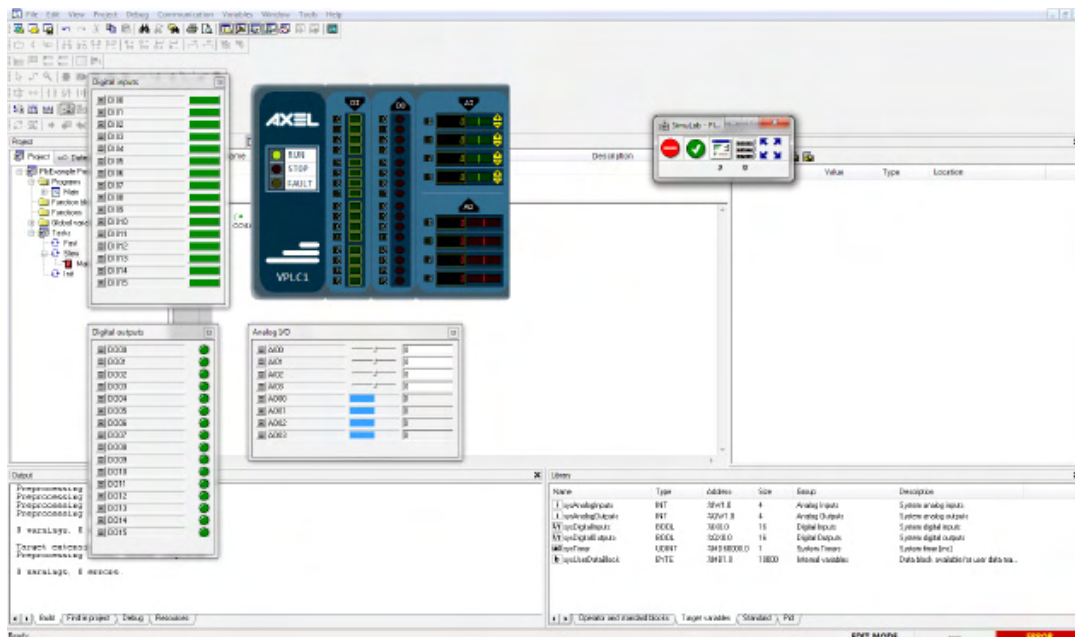
Il codice dell'editor testuale è posizionato nella parte bassa della finestra.

6) Inserire una nuova variabile locale premendo il tasto `Insert record` nella barra `Project`. Altrimenti, è possibile usare l'opzione `Insert` del menu `Variables`.

7) Modificare il nome (per esempio, `Counter`) ed il tipo di variabile. Un doppio click sul campo type apre la finestra di dialogo di selezione del tipo. Scegliere il tipo di dato INT.

8) Implementare un semplice counter in linguaggio ST, come mostrato sotto.

9) Per essere eseguito, il programma `Main` deve essere assegnato ad un task.

Nella finestra `Workspace`, cliccare col tasto destro sul task `Slow` e selezionare l'opzione `Add program`. Si apre la finestra pop-up `Object browser`.

10) Selezionare il programma `Main` e premere `OK`. L'immagine sotto mostra la finestra `Workspace` dopo che l'operazione è stata completata con successo.

11) Selezionare `Simulation mode` dal menu `Debug`.

Rinominate il vostro spazio di lavoro `VPLC1` come PLCExampleWsp e premere `OK`.

12) *VPLC1* è un software che emula un semplice controllo a logica programmabile fornito di digitale e di analogico I/O. E possibile interagire con il simulatore come spiegato nel manuale di SimuLab o brevemente nella scheda *Resources.*

Lo stato del *VPLC1* I/O è disponibile in LogicLab come le *Target variables* elencate nella finestra *Library.*

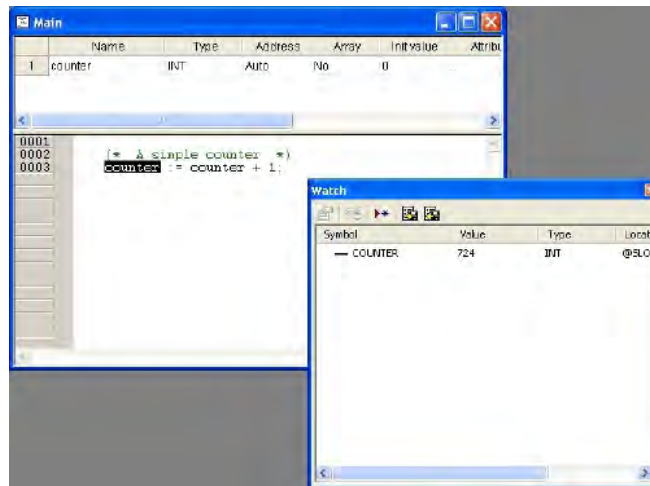
13) Connettere LogicLab a *VPLC1* tramite l'apposito tasto nella barra d'applicazione.

14) Premere *F7* (o selezionare l'opzione *Project-Compile* dal menu) per compilare l'applicazione. Verificare che la compilazione finisca con *0 warnings - 0 errors*, altrimenti correggere gli errori e compilare nuovamente.

15) Eseguire il download del codice mediante l'apposito comando sulla barra delle applicazioni.

Il PLC ora è in esecuzione su *VPLC1.* Potete vedere nella barra in basso di LogicLab l'indicazione *Source ok* - *Connected*.

16) Aprire la finestra *Watch* attraverso il menu *View*, poi selezionare la variabile nell'editor del testo con un doppio click prenderla e trascinarla nella finestra *Watch*.

La finestra *Watch* mostra il valore della variabile *Counter* che dovrebbe incrementare progressivamente.

Siete arrivati alla fine di questa semplice ma completa (modifica-compilazione-download-debug) sessione di lavoro con LogicLab.

Altri passaggi che occorre fare per acquisire una conoscenza più profonda dello strumento sono l'uso di altri linguaggi, blocchi di funzione & funzione, variabili globali, librerie, debug in tempo reale e così via. Le sezioni seguenti di questo manuale vi forniranno le informazioni necessarie.

2.3 UN ESEMPIO PIU' COMPLESSO

Nella sotto directory *SampleProjects\Sample* della vostra installazione LogicLab potete trovare un esempio più complesso di *VPLC1*, che vi offre una panoramica di tutti e cinque i linguaggi IEC 61131-3.

Per aprire il progetto usare l'opzione *File-Open project* del menu e aprire *PLC1Sample.ppjs* o selezionarlo da *Example projects* nella pagina di benvenuto.

3. USARE L'AMBIENTE

Questo capitolo mostra come trattare coi numerosi elementi UI di cui LogicLab è composto, per permettervi di programmare l' IDE nel modo più adatto al vostro specifico progetto di sviluppo.

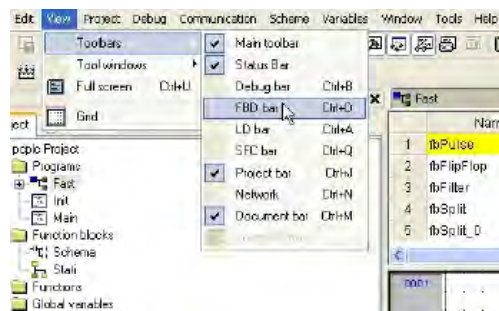
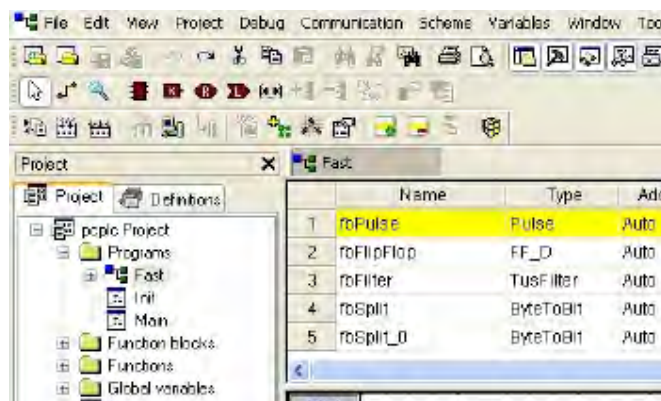
3.1 PERSONALIZZAZIONE DEL LAYOUT

Il layout dello spazio di lavoro di LogicLab può essere liberamente personalizzato secondo le vostre esigenze.

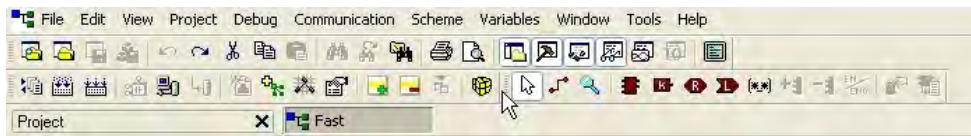
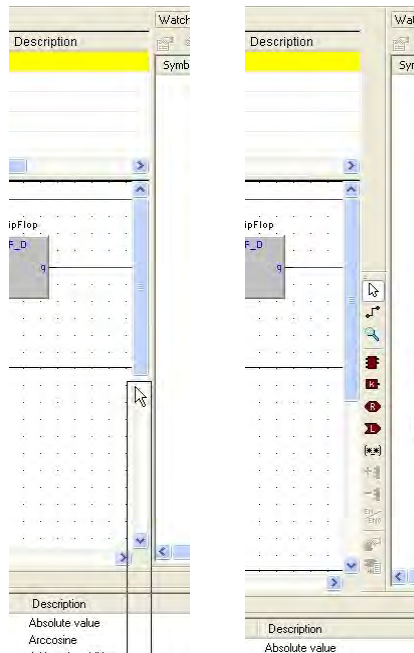
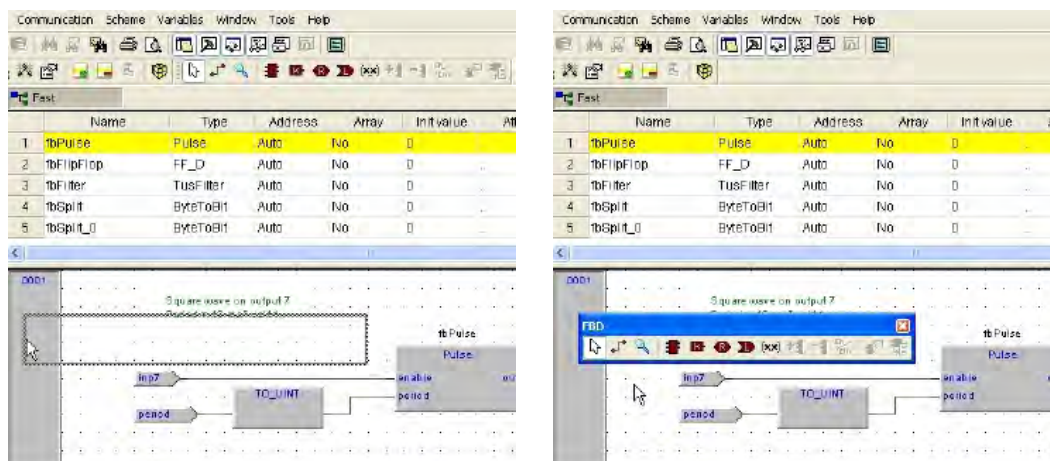
LogicLab si preoccupa di salvare la configurazione del layout all'uscita dall'applicazione, per mantenere le vostre preferenze in sessioni di lavoro differenti.

3.2 TOOLBARS

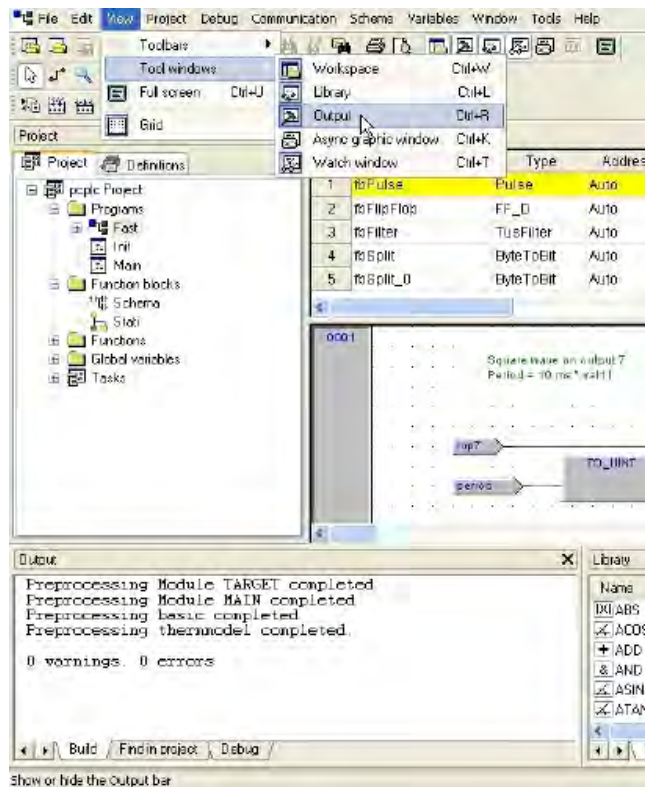
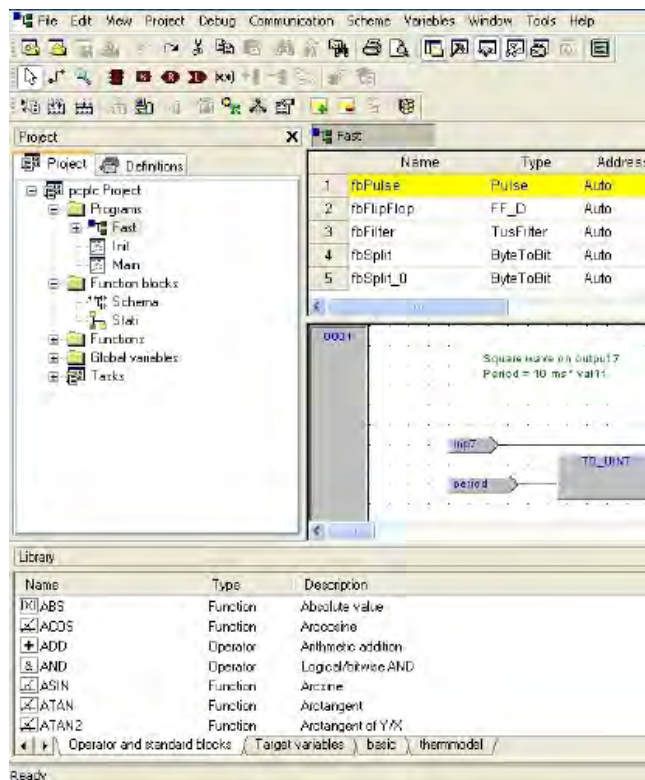
3.2.1 MOSTRARE / NASCONDERE LE BARRE

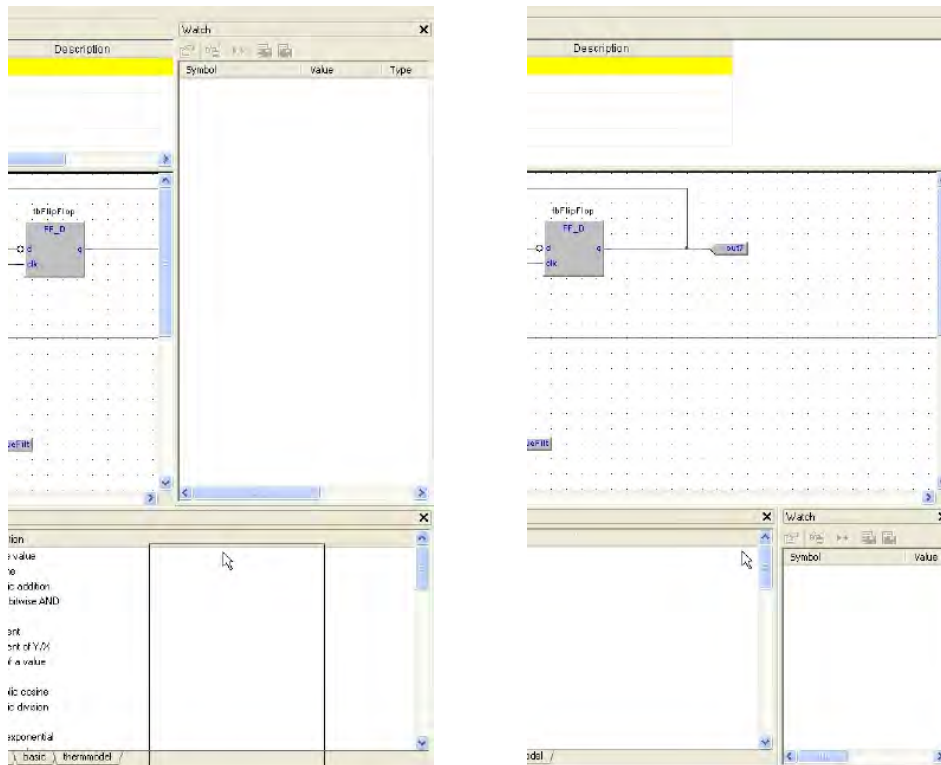
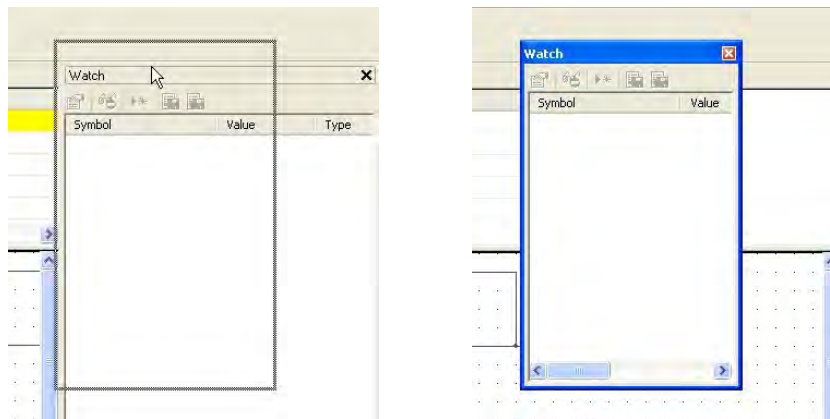
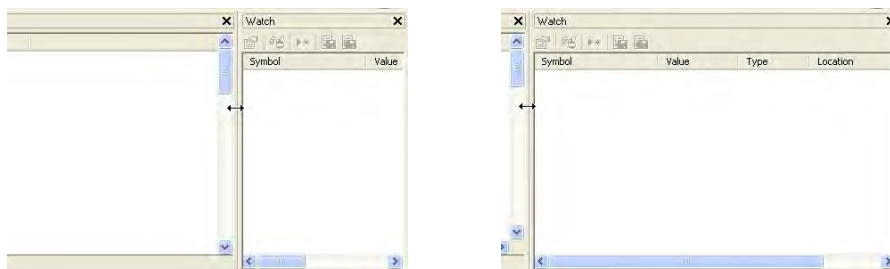
Per mostrare (o nascondere) un barra di applicazione, aprire il menu `View>Toolbars` e selezionare la barra d'applicazione desiderata (per esempio, la barra `Function Block Diagram`).

La barra d'applicazione è ora visibile (nascosta).

3.2.2 SPOSTARE LE BARRE

E' possibile muovere una barra d'applicazione cliccando sulla sua parte sinistra e poi prendendola e trascinandola fino a destinazione.

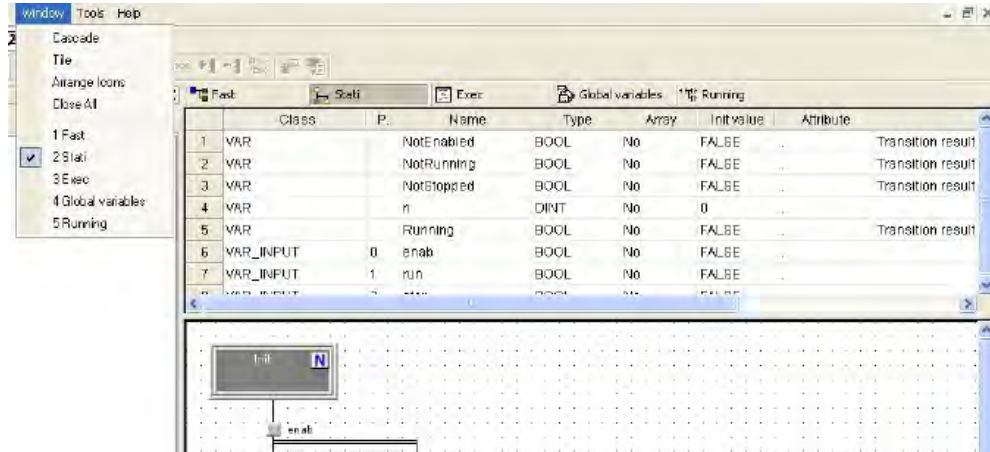
La barra di applicazione compare nella nuova posizione.

E’ inoltre possibile cambiare la forma della barra d’applicazione, da orizzontale a verticale, sia premendo il tasto *Shift* sia spostando la barra vicino al bordo verticale di una finestra qualsiasi.

E’ anche possibile rendere mobile la barra delle applicazioni, sia premendo il tasto *CTRL* sia spostando la barra via da un qualsiasi bordo della finestra.

3.3 LE FINESTRE FISSE

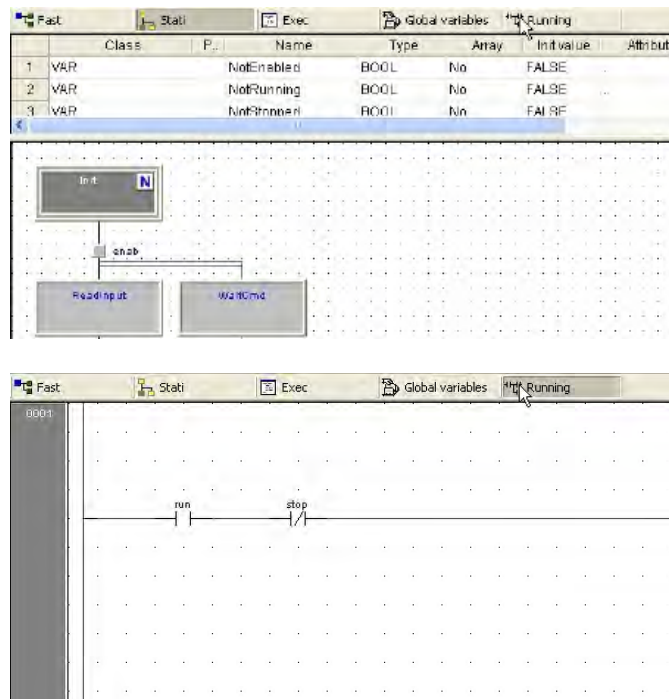
3.3.1 MOSTRARE/NASCONDERE LE FINESTRE DI STRUMENTI

Il menu a finestra *View>Tool* permette di mostrare (o nascondere) una finestra strumento (ad esempio, la finestra *Output*).

La finestra strumento è ora visibile (o nascosta).

3.3.2 SPOSTARE LE FINESTRE DI STRUMENTI

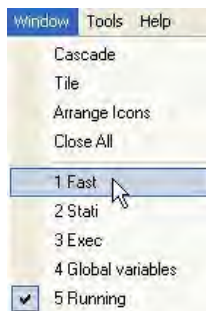
Per muovere una finestra strumento, cliccare sul suo nome (nella parte alta della finestra) e quindi prenderla e trascinarla a destinazione.

E' possibile rendere mobile la finestra strumento cliccando due volte sul suo nome, premendo il tasto *CTRL* o spostando la finestra strumento dai bordi della finestra principale.

Una finestra strumento può essere ridimensionata cliccando e trascinando i suoi bordi fino al raggiungimento della dimensione desiderata.

3.4 LAVORARE CON LE FINESTRE

LogicLab permette di aprire numerosi editor di codice sorgente, così che lo spazio di lavoro potrebbe risultare piuttosto disordinato.

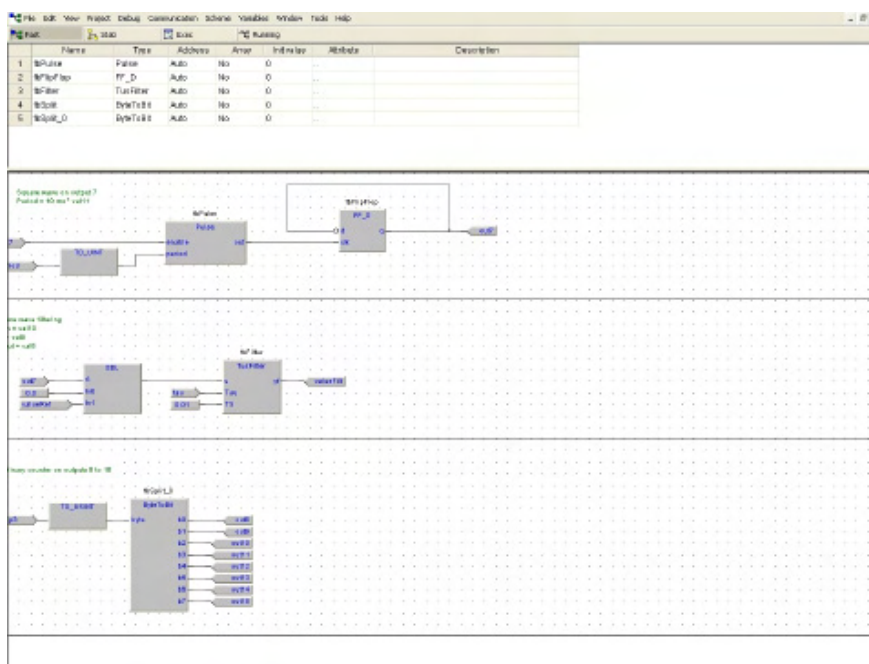
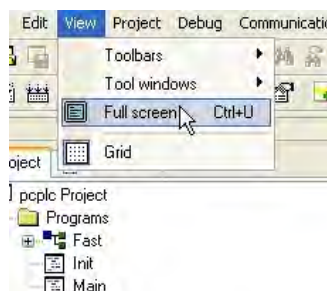
Si può facilmente navigare attraverso queste finestre grazie alla barra *Document* e al menu *Window*.

3.4.1 LA BARRA DOCUMENT

La barra *Document* permette di passare da un editor aperto all'altro, cliccando semplicemente sul nome corrispondente.

Si può mostrare o nascondere la barra *Document* con l'opzione del menu avente lo stesso nome, nel menu *View>Toolbars*.

3.4.2 IL MENU WINDOW

Il menu *Window* è alternativo alla barra *Document*: elenca tutti gli editor aperti e permette di passare dall' uno all'altro.

In aggiunta, questo menu fornisce dei comandi che permettono di rendere automatiche alcune mansioni base, come la chiusura di tutte le finestre.

3.5 MODALITA' SCHERMO INTERO

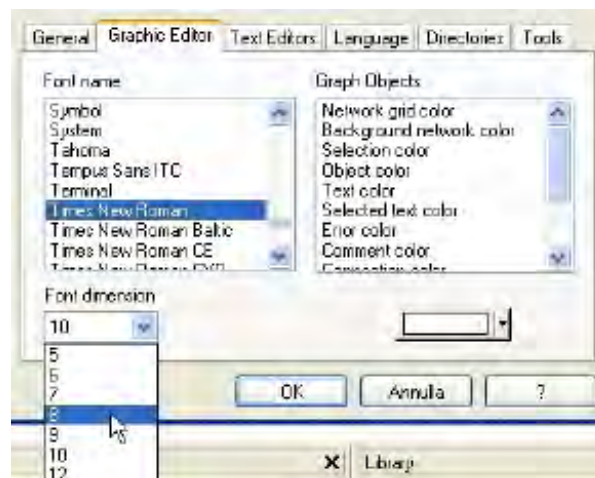
Per facilitare la codifica dell'applicazione, potrebbe essere utile passare alla modalità schermo intero. In questa modalità, l'editor del codice sorgente estende le proprie dimensioni a tutta l'area di lavoro, rendendo più semplice l'attività di modifica del codice, soprattutto quando sono coinvolti linguaggi di programmazione grafici (LD, FBD, e SFC).

E' possibile attivare e disattivare la modalità schermo intero con l'opzione *Full screen* del menu *View* o col comando corrispondente della barra *Main*.

3.6 OPZIONI D'AMBIENTE

Cliccando su *Options..* nel menu *File*, appare una finestra di dialogo multi-tab che permette di personalizzare alcune opzioni di LogicLab.

General

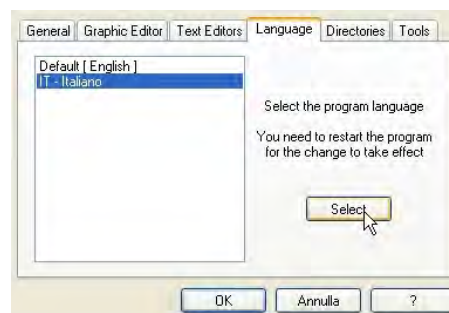
Salvataggio automatico: se la casella *Enable Autosave* è selezionata, LogicLab salva di tanto in tanto l'intero progetto. E' possibile specificare l'intervallo di esecuzione di questa funzione immettendo il numero di minuti che si vuole far intercorrere tra un salvataggio automatico e l'altro nella casella di testo *Autosave interval*.

Graphic Editor

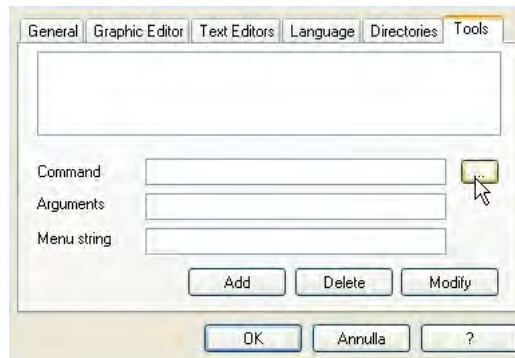
Questo pannello permette di modificare le proprietà degli editor del codice sorgente LD, FBD e SFC.

Text Editors

Language

Si può cambiare la lingua dell'ambiente selezionandone una nuova nella lista mostrata in questo pannello.

Dopo aver selezionato la nuova lingua, premere *Select* e confermare cliccando su *OK*. Il cambiamento sarà effettivo dalla successiva apertura di LogicLab.

Tools

Si possono aggiungere fino a 16 comandi al menu *Tools*. Questi comandi possono essere associati a qualsiasi programma che sarà in funzione sul vostro sistema operativo. E' inoltre possibile specificare un argomento per ogni comando aggiunto al menu *Tools*. La procedura seguente mostra come aggiungere uno strumento al menu.

1) Digitare l'intero percorso del file eseguibile dello strumento nella casella di testo *Command*. Altrimenti, si può specificare il nome del file selezionandolo da Windows Explorer, che si apre cliccando sul tasto *Browse*.

2) Nella casella *Arguments*, digitare gli argomenti – se ne esistono - da trasferire al comando eseguibile menzionato al passaggio 1. Ricordare di separarli con uno spazio.

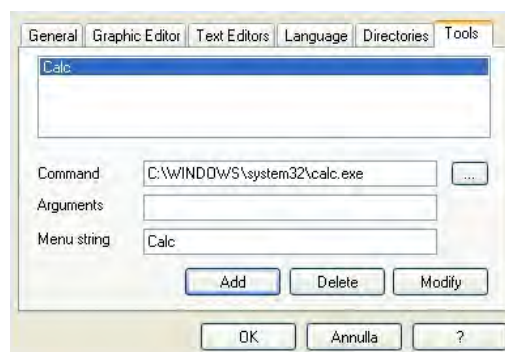
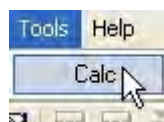
3) Inserire in *Menu string* il nome che si vuole dare allo strumento che si sta aggiungendo. Questa è la stringa che verrà visualizzata nel menu *Tools*.

4) Premere *Add* per inserire effettivamente il nuovo comando nel menu adatto.

5) Premere OK per confermare o Cancel per annullare.

Per esempio, presumiamo che si voglia aggiungere la *calcolatrice* di *Windows* al menu *Tools*:

- Riempire i campi della finestra di dialogo come mostrato.

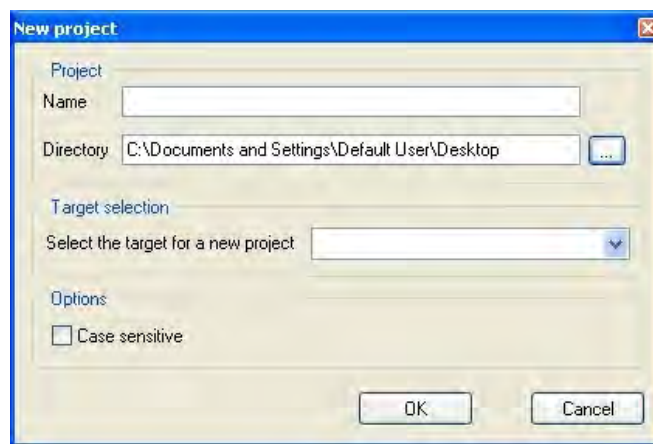
- Premere *Add*. Il nome dato al nuovo strumento è ora visualizzato nella lista di caselle nella parte superiore del pannello

ed anche nel menu *Tool*.

4. GESTIRE I PROGETTI

Questo capitolo si focalizza sui progetti LogicLab.

Un progetto corrisponde ad un'applicazione PLC e include tutti gli elementi richiesti per far funzionare questa applicazione sul target device, compreso il suo codice sorgente, i link alle librerie, le informazioni sul target device e così via.

I paragrafi seguenti spiegano come lavorare propriamente con i progetti e i loro elementi.

4.1 CREARE UN NUOVO PROGETTO

Per iniziare un nuovo progetto, cliccare *New project* nel menu *File* della finestra principale di LogicLab. Lo stesso comando è disponibile nella barra *Main* e, se non è aperto nessun progetto, nella Welcome page di LogicLab. Ciò fa apparire la finestra di dialogo seguente.

E' richiesto l'inserimento del nome del nuovo progetto nel campo *Name*. La stringa che viene inserita sarà anche il nome della cartella che conterrà tutti i file che compongono il progetto LogicLab. Il nome del percorso nel campo *Directory* indica la posizione di default di questa cartella .

Target selection permette di specificare il target device che farà funzionare il progetto.

Per finire, si può rendere il progetto case-sensitive attivando la relativa opzione. Notare che, per default, questa opzione non è attiva, in conformità con lo standard IEC 61131-3: se si sceglie di creare un progetto case-sensitive questo non sarà più conforme allo standard.

Una volta che la decisione di creare un nuovo progetto è stata confermata e le informazioni richieste sono state fornite, LogicLab completa l'operazione, creando la directory e i file del progetto; a questo punto il progetto è aperto.

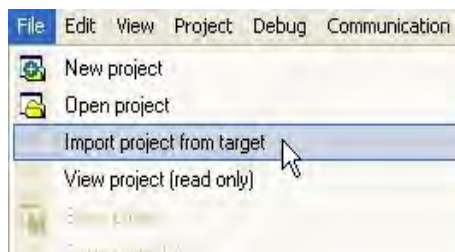
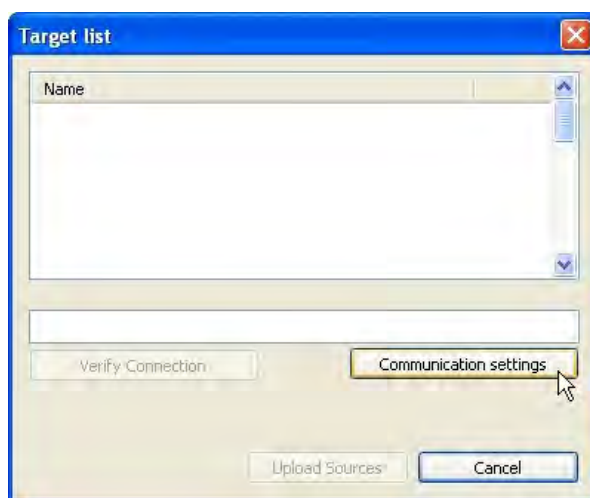
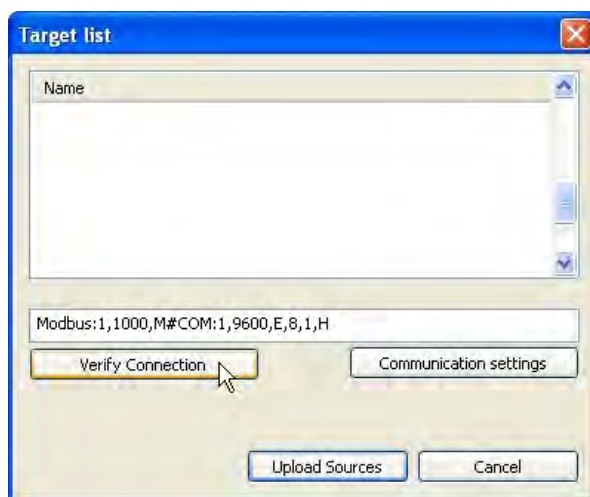
La lista dei device da cui è possibile selezionare il target per il progetto che si sta creando dipende dai contenuti del catalogo dei target device disponibili per LogicLab.

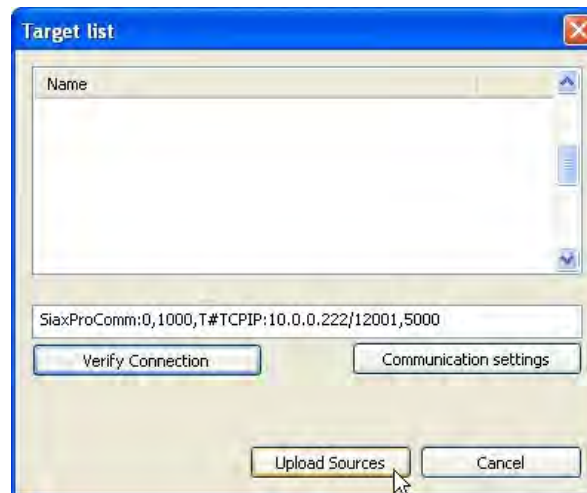
Se manca il target desiderato, o si è lanciato l'eseguibile di setup sbagliato occorre lanciare un setup separato responsabile dell'aggiornamento del catalogo. In tutti e due i casi, è consigliabile contattare il fornitore dell'hardware per avere supporto.

4.2 CARICARE IL PROGETTO DAL SISTEMA TARGET

A seconda del target device con cui ci si sta interfacciando, potete caricare un progetto di lavoro di LogicLab direttamente dal target.

Per caricare il progetto dal target device, seguire la seguente procedura:

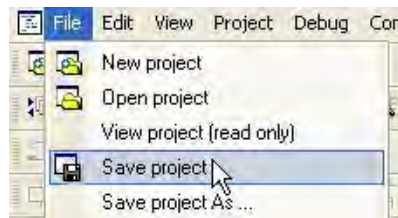
1) Selezionare la voce `Import project from target` nel menu `File`.

2) Selezionare il target device da connettere dalla lista mostrata nella finestra `Target list`.
3) Impostare la comunicazione (per dettagli vedere la sezione Impostare la comunicazione).

4) E' possibile testare la connessione con il target device.

LogicLab prova ad aprire la connessione e riporta il risultato del test.

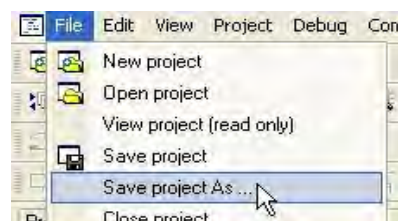
5) Confermare l'operazione.

Quando l'applicazione è caricata correttamente, il progetto viene aperto per le modifiche.

4.3 SALVARE IL PROGETTO

4.3.1 MANTENERE LE MODIFICHE AL PROGETTO

Quando si apporta una qualsiasi modifica al progetto (per esempio, si aggiunge una nuova Program Organization Unit) occorre salvare il progetto per mantenere il cambiamento.

Per salvare il progetto, è possibile selezionare la voce corrispondente del menu `File` o della barra `Main`.

4.3.2 SALVARE IN UNA DIVERSA LOCAZIONE

Quando non si vuole (o non si può, ad esempio perché il file è in sola lettura) riscrivere il file progetto, si può salvare la versione modificata del progetto in una posizione alternativa, selezionando `Save project as…` dal menu `File`.

LogicLab richiede di selezionare la nuova destinazione (che deve essere una directory vuota), poi salva una copia del progetto in quella posizione e apre il file del nuovo progetto per eventuali modifiche.

4.4 GESTIRE I PROGETTI ESISTENTI

4.4.1 APRIRE UN PROGETTO ESISTENTE DI LOGICLAB

Per aprire un progetto esistente, cliccare su `Open project` nel menu `File` della finestra LogicLab principale, o nella barra `Main`, o nella `Welcome page` (quando non è aperto nessun progetto). Ciò fa apparire una finestra di dialogo, che permette di caricare la directory contenente il progetto e selezionare il file di progetto relativo.

4.4.2 MODIFICARE IL PROGETTO

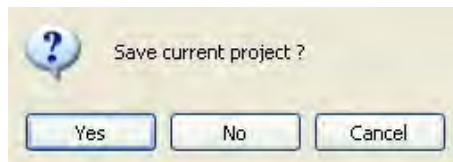
Per modificare un elemento di un progetto, bisogna prima aprire questo elemento cliccando due volte sul suo nome, che si può trovare navigando la struttura ad albero della linguetta di navigazione del progetto, sulla barra `Workspace`.

Cliccando due volte sul nome dell'oggetto che si vuole modificare, si apre un editor coerente con il tipo di oggetto : per esempio, quando si clicca due volte sul nome di una POU di un progetto, viene mostrato l'appropriato editor del codice sorgente; cliccando due volte il nome di una variabile locale, viene mostrato l'editor delle variabili.

E' importante notare che LogicLab impedisce di applicare modifiche agli elementi di un progetto, quando persiste almeno una delle seguenti condizioni:

- Non è possibile modificare alcun oggetto del progetto in modalità debug
- Non è possibile modificare un oggetto di una libreria inclusa, mentre si può modificare un oggetto importato da una libreria.
- Il progetto è aperto in modalità sola lettura (vedere progetto).

4.4.3 CHIUDERE IL PROGETTO

Si può terminare la sessione di lavoro sia chiudendo esplicitamente il progetto sia uscendo da LogicLab. In entrambi i casi, quando ci sono modifiche non ancora salvate su file, LogicLab richiede di scegliere se salvare o meno tali modifiche.

Per chiudere il progetto, selezionare la voce `Close project` dal menu `File`; LogicLab mostra la `Welcome page`, permettendo di iniziare rapidamente una nuova sessione di lavoro.

4.5 DISTRIBUIRE I PROGETTI

Quando c'è la necessità di condividere un progetto con un altro sviluppatore è possibile inviare o una copia dei file del progetto o un redistributable source module (RSM) generato da LogicLab.

Nel primo caso, il numero di file che si deve condividere dipende dal formato del file del progetto:

- PLC file singolo di progetto (estensione file *.ppjs*): il file del progetto contiene tutte le informazioni richieste per far funzionare l'applicazione (ammesso che lo sviluppatore ricevente abbia un target device appropriato disponibile) compresi i moduli di codice sorgente, così che occorra condividere solo il file *.ppjs*.
- PLC file multiplo di progetto (estensione file *.ppjx* o *.ppj*): il file di progetto contiene solo i link ai moduli di codice sorgente che compongono il progetto, i quali sono memorizzati come file singoli nella directory del progetto. Occorre condividere l'intera directory.

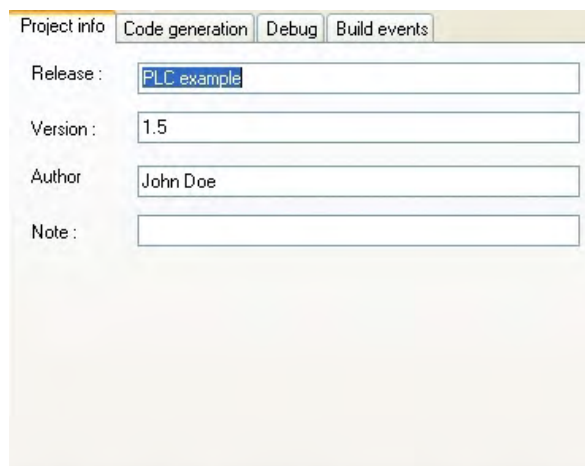
Altrimenti, è possibile generare un modulo di codice sorgente ridistribuibile (RSM) tramite la voce corrispondente del menu *Project* o della barra d'applicazione.

LogicLab notifica il nome del file RSM e permette di scegliere se proteggere il file con una password o meno. Scegliendo di proteggere il file, LogicLab richiederà di inserire la password.

TI vantaggi del formato del file RSM sono:

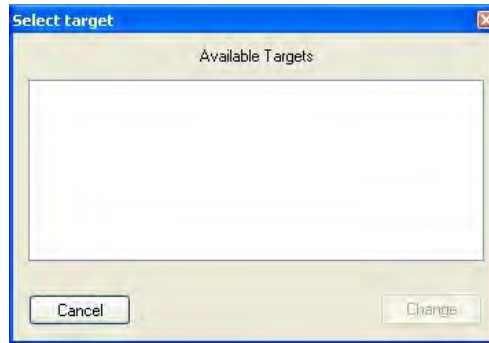
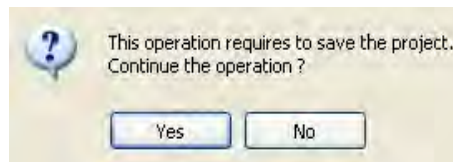
- il codice sorgente è codificato in codice binario, quindi non può essere letto da terze parti che non utilizzano LogicLab, garantendo un trasferimento in rete più sicuro;
- può essere protetto con una password, che verrà richiesta da LogicLab al momento dell'apertura del file;
- trattandosi di un file binario, la sua dimensione è ridotta.

4.6 OPZIONI DEL PROGETTO

E' possibile modificare alcune proprietà basiche del progetto, come il nome e la versione dell'applicazione, nella finestra che appare dopo aver selezionato la voce *Options…* nel menu *Project*.

L'informazione qui inserita sarà visualizzata in ogni documento stampato e potrà essere inoltre scaricata sul target device.

4.7 SELEZIONARE IL SISTEMA TARGET

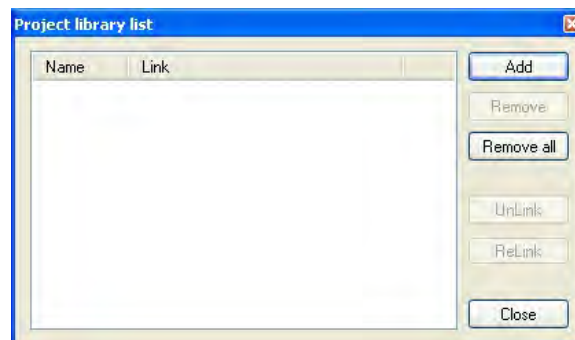
Potrebbe essere necessario trasferire un'applicazione PLC su un target device diverso da quello per cui il codice era stato inizialmente scritto. Seguire le istruzioni sottostanti per adattare il proprio progetto LogicLab al nuovo target device.

1) Cliccare *Select target* nel menu *Project* della finestra principale di LogicLab. Ciò farà apparire la seguente finestra di dialogo.

2) Selezionare uno dei target device elencati nella casella combinata.
3) Cliccare *Change* per confermare, *Cancel* per abbandonare.
4) Confermando, LogicLab mostra la finestra di dialogo seguente.

Premere *Yes* per completare l'operazione, *No* per abbandonare.

Premendo *Yes*, LogicLab farà l'aggiornamento del progetto per farlo funzionare col nuovo target. Farà anche una copia di backup dei file del progetto in una sotto directory inserita nella directory del progetto, in modo che si possa riportare indietro l'operazione rimpiazzando manualmente (per es. usando Windows Explorer) i file del progetto con la copia di backup.

4.8 LAVORARE CON LE LIBRERIE

Le librerie sono un potente strumento per condividere oggetti tra i progetti di LogicLab. Le librerie sono normalmente memorizzate in specifici file sorgente, con estensione *.pll*.

4.8.1 IL GESTORE DELLA LIBRERIA

Il gestore della libreria elenca tutte le librerie incluse al momento in un progetto LogicLab.

Permette inoltre di inserire o rimuovere librerie. Per accedere al gestore della libreria, cliccare *Library manager* nel menu *Project*.

4.8.1.1 INCLUDERE UNA LIBRERIA

La seguente procedura mostra come inserire una libreria in un progetto LogicLab che risulti in tutti gli oggetti della libreria divenuti disponibili per il progetto corrente.

Inserire una libreria significa che viene aggiunto al progetto corrente un riferimento al file *.pll* della libreria, e che viene fatta una copia locale della libreria. E' opportuno notare che non è possibile modificare gli elementi di una libreria inserita, a differenza degli oggetti importati.

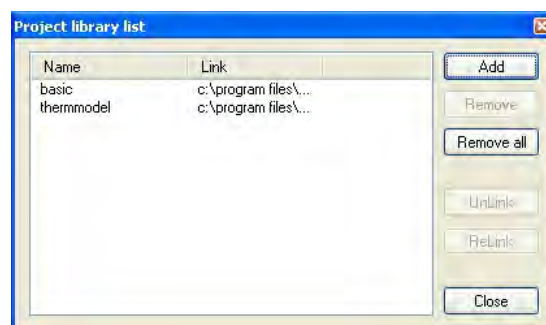
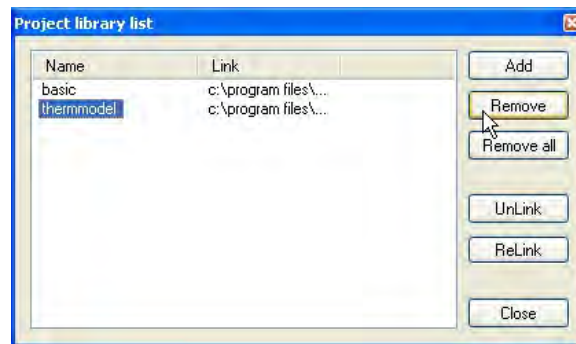
Se si vuole copiare o spostare un progetto che include una o più librerie, accertarsi che i riferimenti a queste librerie siano sempre validi nella nuova posizione.

1) Cliccare *Library manager* nel menu *Project*, aprendo così la finestra di dialogo *Library manager*.
2) Premere il tasto *Add*, che farà apparire una finestra di dialogo esplorativa per permettere di selezionare il file *.pll* della libreria che si desidera aprire.
3) Una volta trovato il file *.pll*, aprirlo tramite il doppio click oppure premendo il tasto *Open*. Il nome della libreria e il suo percorso sono ora visualizzati in una nuova riga alla fine dell'elenco della casella bianca.

4) Ripetere i passaggi 1,2 e 3 per ogni libreria che si desidera inserire.
5) Una volta terminato l'inserimento, premere *Ok* per confermare o *Cancel* per abbandonare.

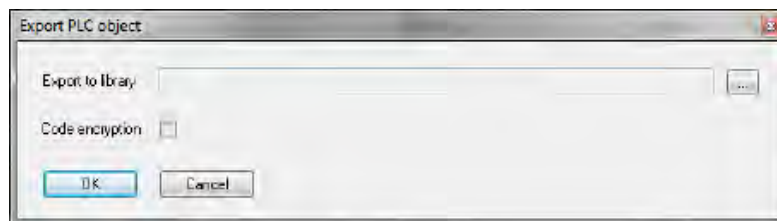
4.8.1.2 RIMUOVERE UNA LIBRERIA

La procedura seguente mostra come rimuovere una libreria inserita dal progetto corrente. Bisogna ricordare che rimuovere la libreria non significa cancellarla, ma eliminare il riferimento del progetto ad essa.

1) Cliccare *Library Manager* nel menu *Project* della finestra principale di LogicLab; si aprirà la finestra di dialogo *Library manager*.

2) Selezionare la libreria che si desidera rimuovere cliccando una volta sul suo nome. Il tasto *Remove* è ora abilitato.

3) Cliccare il tasto *Remove*; il riferimento alla libreria selezionata sparirà dall'elenco *Project library*.
4) Ripetere l'operazione per tutte le librerie che si desidera rimuovere. Altrimenti, per rimuoverle tutte, cliccare su *Remove all*.
5) Una volta finito di rimuovere le librerie, premere *OK* per confermare o *Cancel* per non apportare nessuna modifica.

4.8.2 ESPORTARE AD UNA LIBRERIA

E' possibile esportare un oggetto da un progetto correntemente aperto ad una libreria, per rendere tale oggetto disponibile per altri progetti. La procedura seguente mostra come esportare oggetti ad una libreria.

1) Cercare l'oggetto che si vuole esportare navigando la struttura ad albero della scheda progetto della barra *Workspace*, poi cliccare una volta sul nome dell'oggetto.
2) Cliccare *Export object* to *library* nel menu *Project*. Ciò farà apparire la finestra di dialogo seguente.

3) Inserire la libreria di destinazione specificando la posizione del suo file .*pll*. Si può fare:
   - Digitando l'intero nome del percorso nel casella di testo bianca;
   - cliccando il tasto *Browse*, per aprire una finestra di dialogo esplorativo che permette di navigare il disco e la rete.
4) Si può scegliere di criptare il codice sorgente della POU che si sta esportando, per proteggere la propria proprietà intellettuale.
5) Premere *Ok* per confermare l'operazione, altrimenti premere *Cancel* per abbandonare.

Se nel passaggio 3 della procedura si inserisce il nome di un file .*pll* non esistente, LogicLab crea il file, creando quindi anche una nuova libreria.

4.8.2.1 CANCELLARE L'ESPORTAZIONE AD UNA LIBRERIA

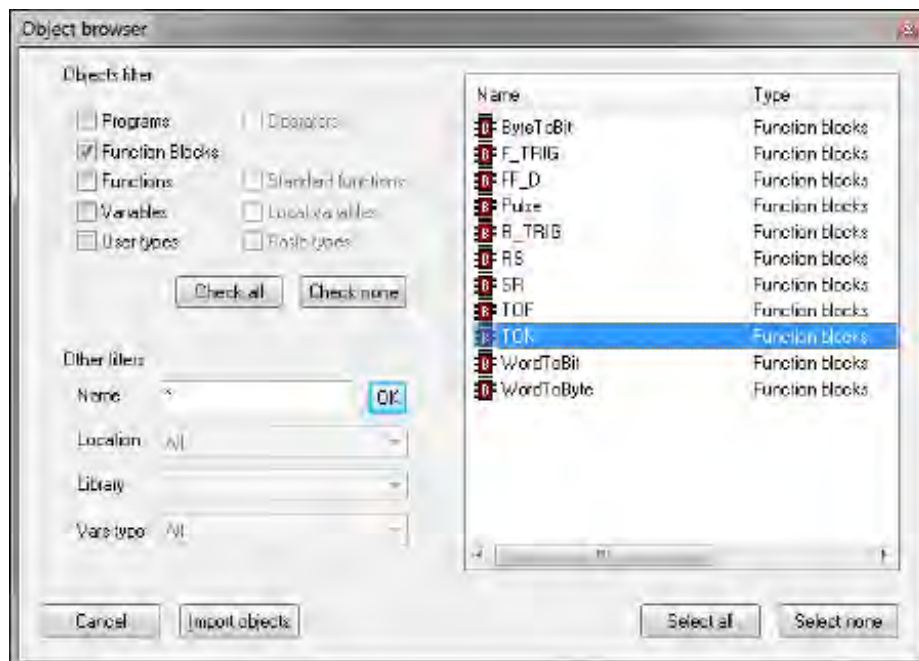
Al momento, non è possibile annullare l'esportazione ad una libreria. L'unico modo per rimuovere un oggetto è creare un'altra libreria contenente tutti gli oggetti di quella corrente, meno l'oggetto che si vuole eliminare.

4.8.3 IMPORTARE DA UNA LIBRERIA O UN'ALTRA FONTE

E' possibile importare un oggetto da una libreria per usarlo nel progetto corrente. Quando si importa un progetto da una libreria, la copia locale dell'oggetto perde il riferimento alla libreria originale e appartiene esclusivamente al progetto corrente. Per questo motivo, è possibile modificare oggetti importati, a differenza degli oggetti delle librerie inserite.

Ci sono due modi per ottenere una POU da una libreria. La procedura seguente mostra come importare oggetti da una libreria.

1) Cliccare *Import object from library* dal menu *Project*. Ciò farà apparire una finestra di dialogo esplorativa che permette di selezionare il file *.pll* della libreria che si vuole aprire.

2) Una volta trovato il file *.pll*, aprirlo col doppio click o premendo il tasto *Open*. La finestra di dialogo esplorativa della libreria apparirà in primo piano. Ciascuna scheda della finestra contiene una lista di oggetti di un tipo coerente con il titolo della scheda.

3) Selezionare la scheda del tipo di oggetto che si vuole importare. E' anche possibile svolgere una semplice ricerca degli oggetti di ogni scheda usando i *Filters*. Notare comunque che alle librerie è applicabile solo il filtro *Name*. Per usarlo, selezionare una scheda, poi immettere il nome dell'oggetto desiderato, usando anche il jolly *, se necessario.

4) Selezionare l'oggetto che si vuole importare, poi premere il tasto *Import object*.

5) Una volta finito di importare oggetti, premere indifferentemente *OK* o *Cancel* per chiudere la finestra di navigazione della *Library*.

4.8.3.1 CANCELLARE L'IMPORTAZIONE DA UNA LIBRERIA

Quando si vuole importare un oggetto in un progetto LogicLab, non si fa altro che creare una copia locale di quell'oggetto. Per questo motivo, è sufficiente cancellare l'oggetto locale per cancellare l'importazione da una libreria.

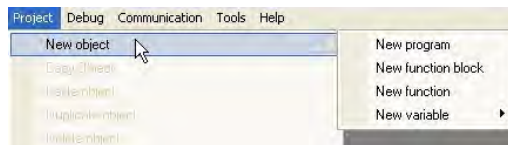
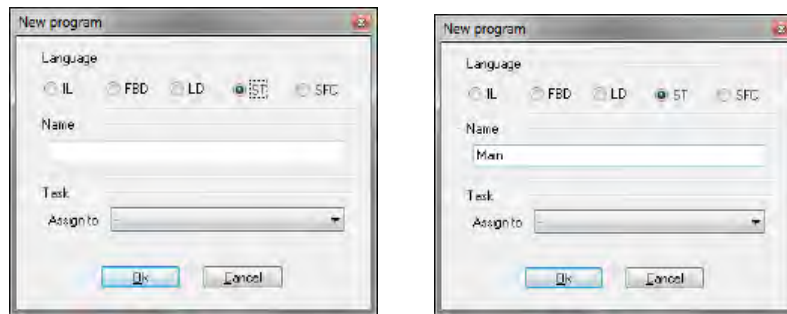
5. GESTIRE GLI ELEMENTI DEL PROGETTO

Questo capitolo mostra come relazionarsi con gli elementi che compongono un progetto, ovvero: Program Organization Units (POUs), task, tipi di dati derivati, e variabili.

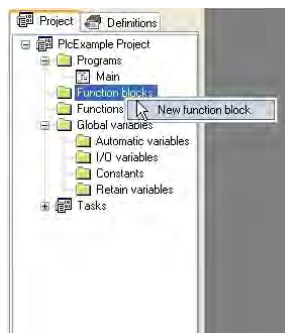
5.1 PROGRAM ORGANIZATION UNITS

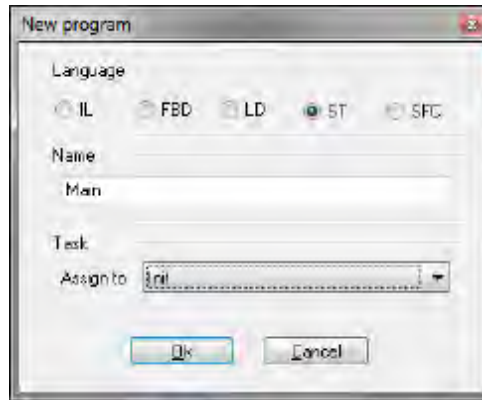
Questo paragrafo mostra come aggiungere nuove POUs al progetto, come modificarle e rimuoverle.

5.1.1 CREARE UNA NUOVA PROGRAM ORGANIZATION UNIT

1) Selezionare la voce *New object* dal menu *Project*..

2) Specificare quale tipo di POU si vuole creare cliccando su una delle voci del sotto menu che appare.
3) Selezionare il linguaggio che si vuole usare per implementare la POU.

Inserire il nome del nuovo modulo.

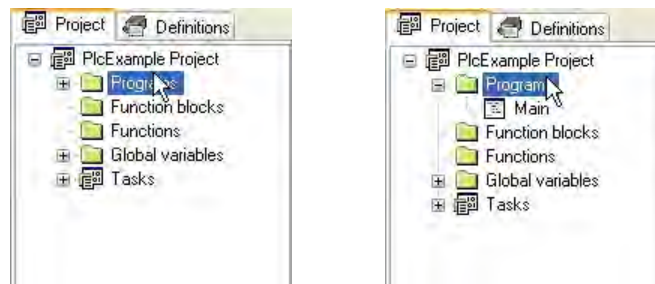
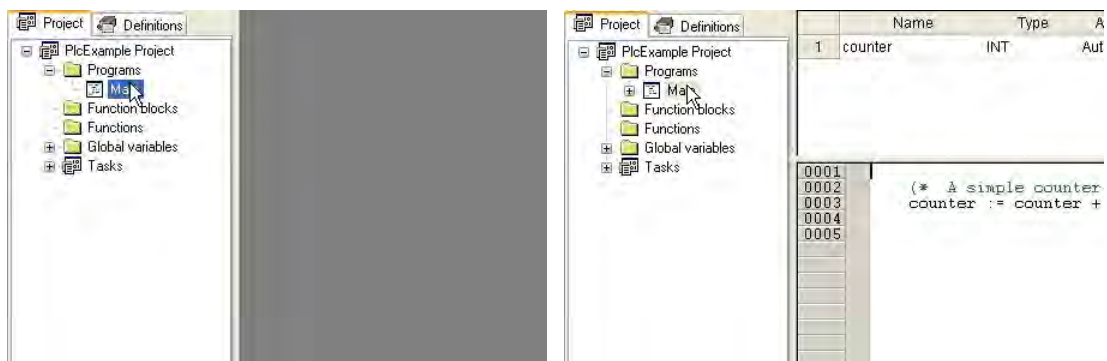
4) Confermare l'operazione premendo *OK*.

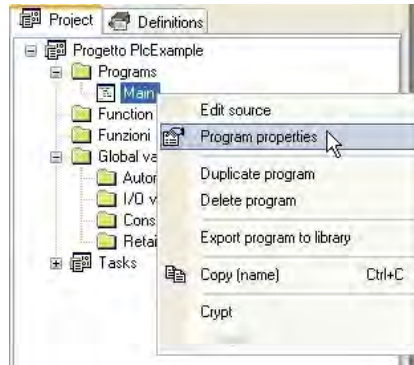
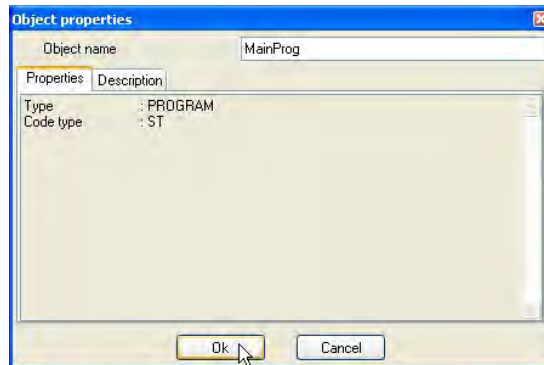
Altrimenti si può creare una nuova POU di un tipo specifico (programma, blocco funzione o funzione) cliccando col tasto destro sulla voce corrispondente dell'albero del progetto.

5.1.1.1 ASSEGNAZIONE DI UN PROGRAMMA AD UN TASK IN FASE DI CREAZIONE

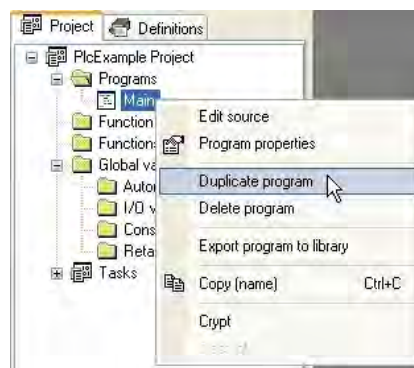
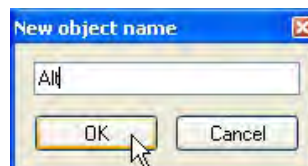
LogicLab dà la possibilità di assegnare un nuovo programma ad un task contestualmente alla sua creazione: selezionare il task a cui assegnare il programma dalla lista mostrata nella selezione *Task* della finestra *New program.*

5.1.2 MODIFICARE LE POUS

Tutte le POU di un progetto sono elencate nelle cartelle *Programs*, *Function blocks* e *Function* della scheda *Project* della barra *Workspace*.

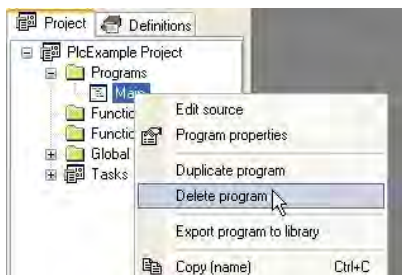
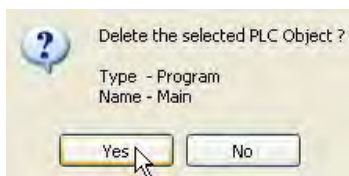
La procedura seguente mostra come modificare il codice sorgente di una POU esistente.

1) Aprire la cartella nella scheda Project dello spazio di lavoro che contiene l'oggetto che si vuole modificare cliccando due volte sul nome della cartella.

2) Cliccare due volte sul nome dell'oggetto che si vuole modificare. L'editor relativo si aprirà permettendo di modificare il codice sorgente della POU.

Per cambiare il nome della POU:

1) Aprire l'editor *Object properties* del menu contestuale che appare cliccando col tasto destro sul nome della POU nell'albero del progetto (altrimenti, selezionare la voce del menu *Project* corrispondente).

2) Modificare il nome dell'oggetto e confermare.

Infine, potete creare una copia della POU nel seguente modo:

1) Selezionare *Duplicate* dal menu contestuale (o dal menu *Project*).

2) Inserire il nome della nuova POU e confermare.

5.1.3 CANCELLARE LE POU

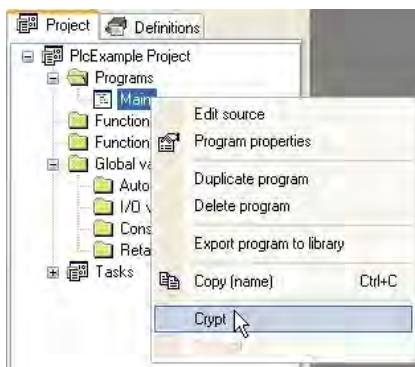
Seguire la seguente procedura per rimuovere una POU dal vostro progetto:

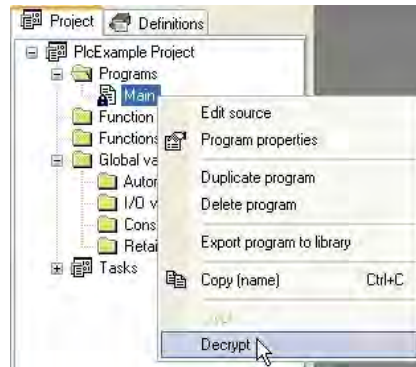
1) Dalla casella *Project* dello spazio di lavoro aprire la cartella contenente l'oggetto che si vuole cancellare cliccando due volte sul nome della cartella.
2) Cliccare col tasto destro sul nome dell'oggetto che si desidera cancellare. Apparirà un menu contestuale riferito all'oggetto selezionato.

3) Cliccare su *Delete object* nel menu contestuale, poi confermare l'operazione premendo *Yes*.

5.1.4 CRIPTARE IL CODICE SORGENTE

Potreste voler nascondere il codice sorgente di una o più POU.

LogicLab vi permette di criptare le POU e proteggerle con una password.

Per criptare una POU, seguire i seguenti passaggi:

1) Cliccare col tasto destro sul nome della POU nell'albero del progetto e scegliere *Crypt* dal menu contestuale..

2) Inserire due volte la password (per evitare i problemi che potrebbero sorgere da errori di battitura) e confermare l'operazione.

Per decriptare una POU, cliccare col tasto destro sul nome della POU nell'albero del progetto e scegliere *Decrypt* dal menu contestuale.

LogicLab vi richiederà di inserire la password.

E' possibile scegliere di criptare tutte le POU non criptate con una sola operazione:

la stessa password sarà valida per tutti gli oggetti.

5.2 VARIABILI

Esistono due classi di variabili in LogicLab: variabili globali e variabili locali.

Questo paragrafo mostra come aggiungere al progetto, modificare e rimuovere sia le variabili globali che quelle locali.

5.2.1 VARIABILI GLOBALI

Le variabili globali possono essere visualizzate e consultate da qualsiasi modulo del progetto.

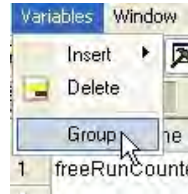
5.2.1.1 CLASSI DI VARIABILI GLOBALI

Le variabili globali sono elencate nell'albero del progetto, nella cartella *Global variables*, dove sono ulteriormente classificate, secondo le loro proprietà, in variabili Automatiche, variabili Mapped , Costanti e variabili Retain.

- Le variabili Automatiche includono tutte le variabili che il compiler alloca automaticamente in una posizione appropriata della memoria del target device.
- Le variabili Mapped, invece, hanno un indirizzo assegnato nel sistema logico di indirizzo del target device, che deve essere specificato dallo sviluppatore.
- Le variabili Costanti sono quelle che lo sviluppatore dichiara come aventi l'attributo `CONSTANT`, così che non possano essere scritte.
- Le variabili Retain sono quelle che lo sviluppatore indica come aventi l' attributo `RETAIN`, cosicchè i loro valori sono memorizzati in un'area della memoria persistente del target device.

5.2.1.2 GRUPPI DI VARIABILI GLOBALI

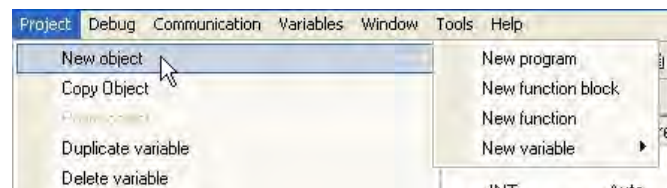
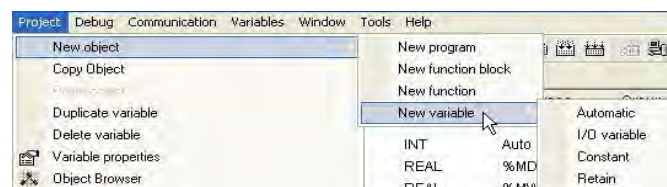
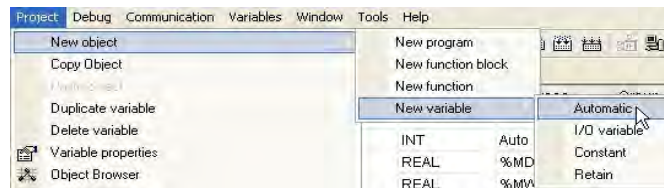
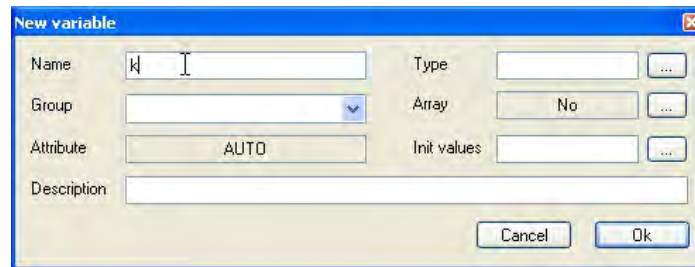
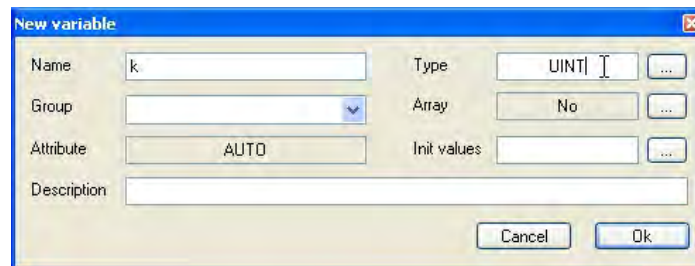
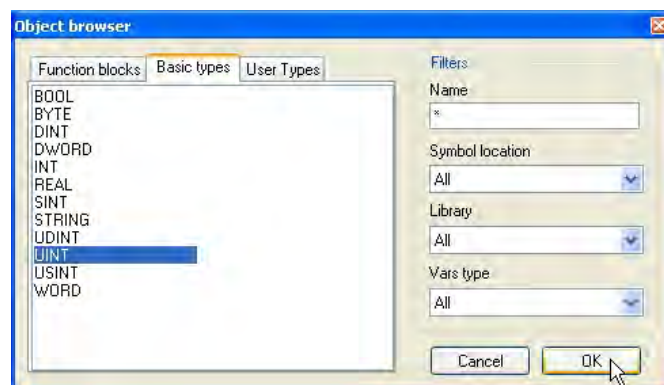
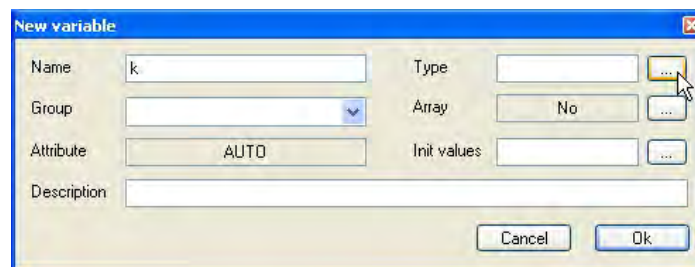
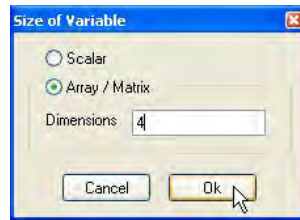
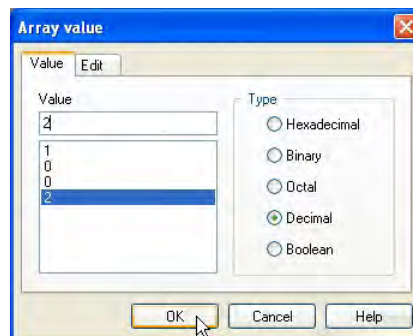
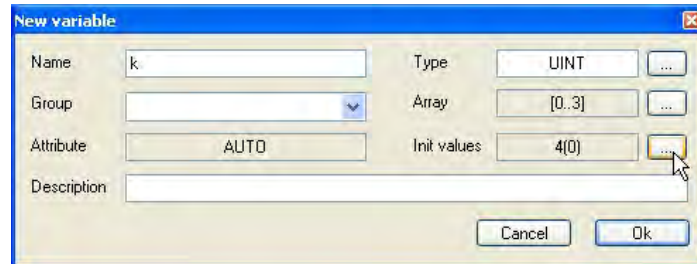
E' possibile categorizzare ulteriormente l'insieme di varianti globali raggruppandole secondo criteri specifici di applicazione. Per definire un nuovo gruppo, seguire la seguente procedura:

1) Selezionare `Group` dal menu `Variables` (notare che questo menu è disponibile solo se l'editor `Global variables` è aperto).

2) Inserire il nome del nuovo gruppo di variabili e premere `Add`.

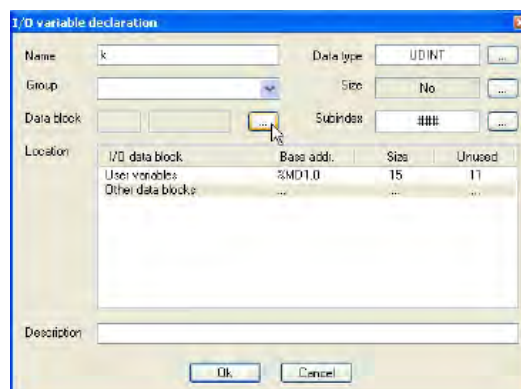
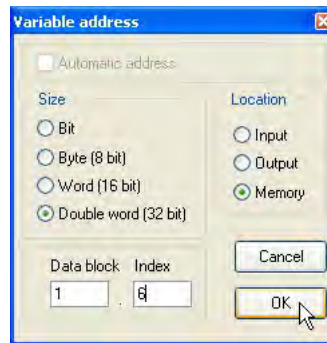
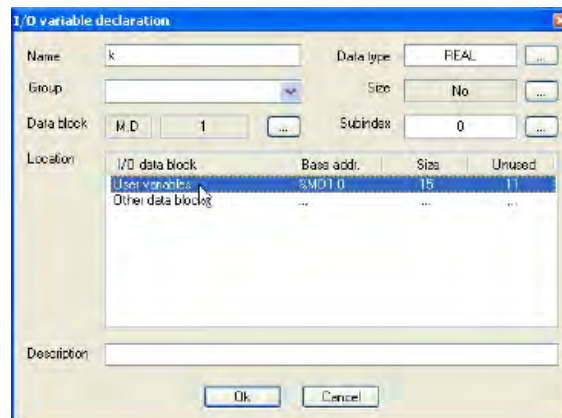
3) E' ora possibile usare il gruppo di variabili nella dichiarazione di nuove variabili globali.

5.2.1.3 CREARE UNA NUOVA VARIABILE GLOBALE

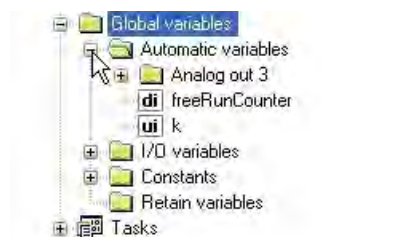
Applicare la seguente procedura per avere una nuova variabile globale:

1) Selezionare `New object` dal `Project` menu.

2) Selezionare `New variable` dal menu che appare.

3) Scegliere la classe della variabile che si vuole creare (variabile Automatica, variabile Mapped, Costante o variabile Retain).

4) Inserire il nome della variabile (tenere presente che i caratteri '?', '.', '/' e così via non possono essere utilizzati: la variabile deve avere un identificatore IEC 61131-3 valido).

5) Specificare il tipo di variabile digitandolo

o selezionandolo dall'elenco che LogicLab mostra quando si clicca sul tasto `Browse`.

6) Se si vuole dichiarare un array, si può specificarne la misura.

7) E' inoltre possibile assegnare il valore iniziale alla variabile.

8) Infine, si può aggiungere una breve descrizione e poi confermare l'operazione.

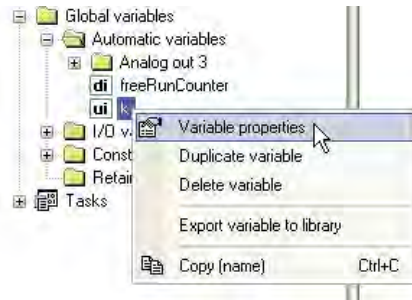
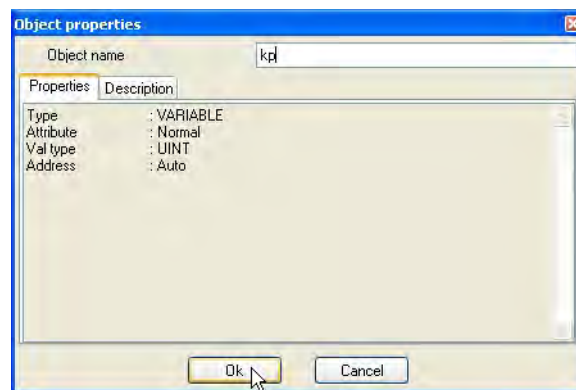
Se si crea una variabile Mapped, si dovrà specificare l'indirizzo della variabile durante la sua definizione. Per fare ciò, seguire le indicazioni:

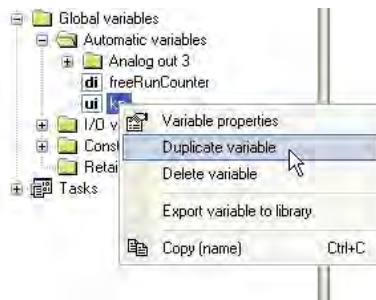
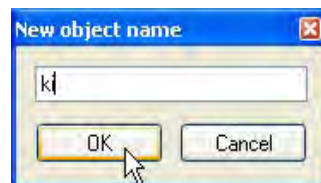
- Cliccare sul tasto per aprire l'editor dell'indirizzo e inserire il valore desiderato.

- Selezionare dall'elenco che LogicLab mostra l'area di memoria che si vuole usare: lo strumento sceglierà automaticamente l'indirizzo della prima posizione libera di memoria di quest'area.

5.2.1.4 MODIFICARE UNA VARIABILE GLOBALE

Per modificare una variabile globale esistente:

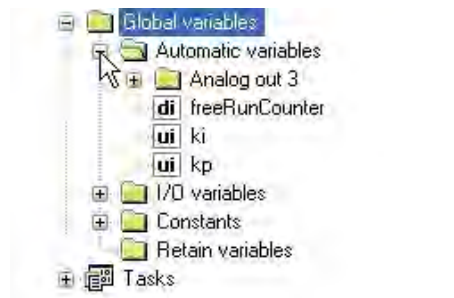
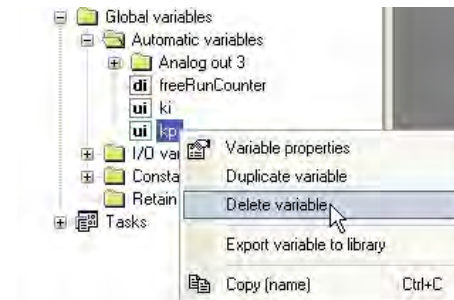
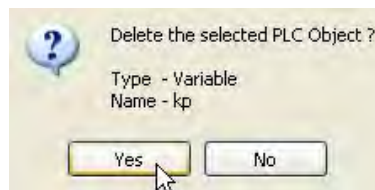
1) Aprire la cartella che contiene la variabile che si desidera modificare dalla casella `Project` dello spazio di lavoro.

2) Cliccare due volte sul nome della variabile da modificare: l'editor delle variabili globali si aprirà, permettendo la modifica.

Per cambiare solamente il nome della variabile:

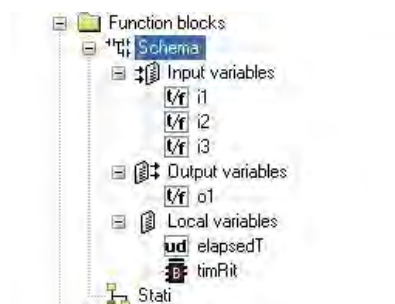
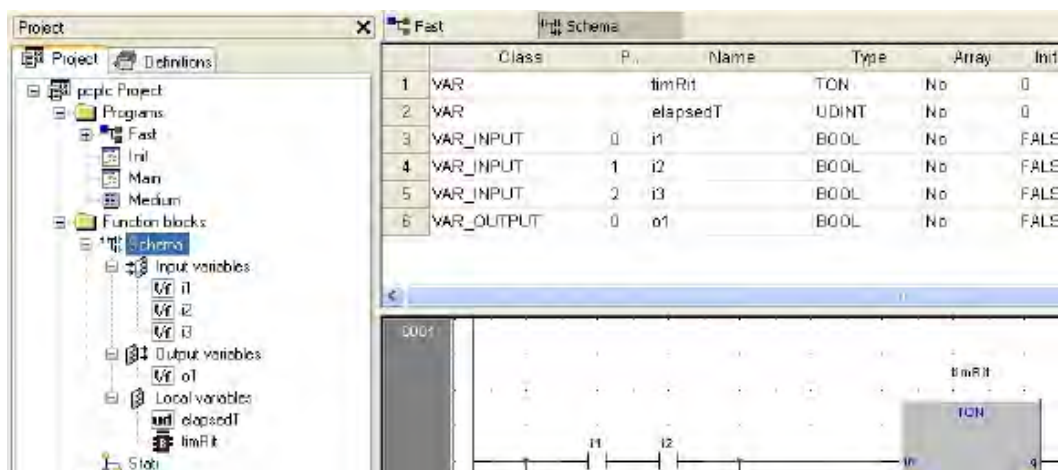
1) Aprire l'editor *Variable properties* dal menu contestuale che appare cliccando col tasto destro sul nome della variabile nell'albero del progetto (o selezionare la voce corrispondente del menu *Project*).

2) Cambiare il nome della variabile e confermare.

Infine, si può creare un duplicato della variabile in questo modo:

1) Selezionare *Duplicate variable* dal menu contestuale (o dal menu *Project*).

2) Inserire il nome della nuova variabile e confermare.

5.2.1.5 CANCELLARE UNA VARIABILE GLOBALE

Seguire la seguente procedura per eliminare una variabile globale dal proprio progetto:

1) Aprire la cartella che contiene la variabile che si vuole eliminare dalla casella *Projec* dello spazio di lavoro.

2) Cliccare col tasto destro sul nome della variabile da cancellare. Apparirà un menu contestuale riferito alla variabile selezionata.

3) Cliccare su `Delete variable` nel menu contestuale, poi premere `Yes` per confermare.

5.2.2 VARIABILI LOCALI

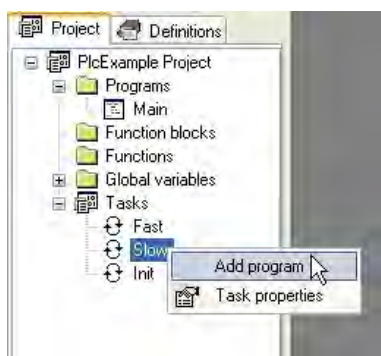
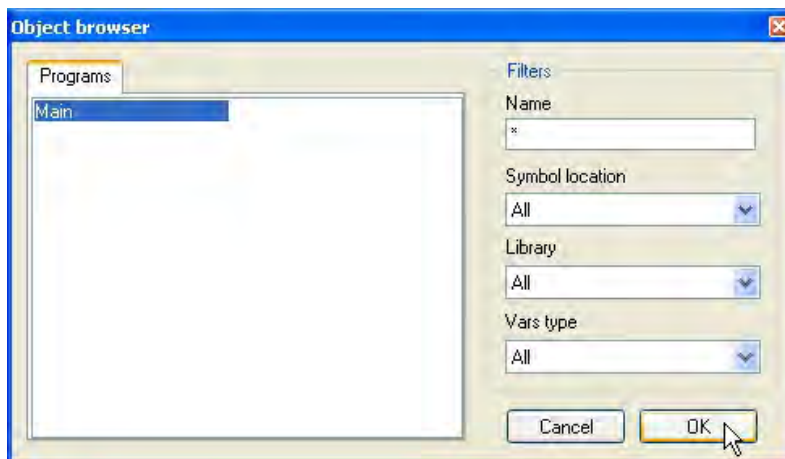
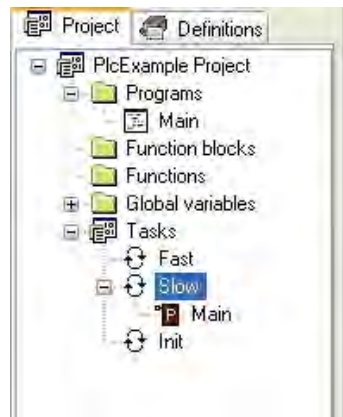
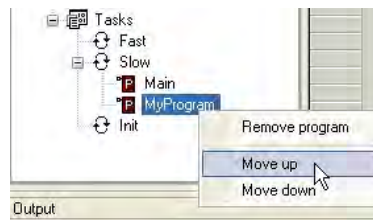
Le variabili locali sono dichiarate all'interno di una POU (programma, funzione o blocco funzione), il modulo è l'unico elemento del progetto che può riferirsi ed accedere ad esse.

Le variabili locali sono elencate nell'albero del progetto sotto alla POU che le dichiara (solo quando la suddetta POU è aperta per la modifica), dove sono ulteriormente classificate secondo la classe a cui appartengono (per esempio variabili input o inout).

Per creare, modificare e cancellare le variabili locali occorre aprire la POU per modificare e usare l'editor variabili locali.

Fare riferimento alla sezione corrispondente di questo manuale per ulteriori dettagli (vedi paragrafo 6.6.1.2).

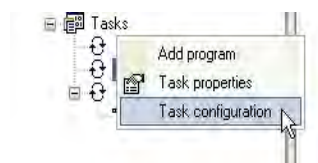
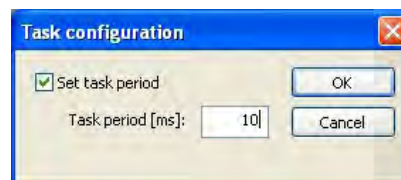
5.3 TASK

5.3.1 ASSEGNARE UN PROGRAMMA AD UN TASK

Leggere le istruzioni sottostanti per sapere come fare eseguire un programma ad un determinato task.

1) I task a disposizione sono elencati nella casella Project della finestra Workspace. Cliccare col tasto destro sul nome del task che si vuole che esegua il programma e scegliere *Add program* dal menu contestuale.

2) Selezionare dalla lista che appare il programma che si vuole far eseguire al task e confermare.

3) Il programma è ora assegnato al task, come si può vedere nell'albero del progetto.

Notare che è possibile assegnare più di un programma ad una mansione. Dal menu contestuale si può classificare ed, in caso, rimuovere i programmi assegnati alle mansioni.

5.3.2 CONFIGURAZIONE DEI TASK

A seconda del target device con cui ci si sta interfacciando, potrebbe essere possibile configurare le impostazioni dei task del PLC.

1) Selezionare la voce `Task configuration` dal menu contestuale che appare nella finestra di pop-up se premente con il tasto destro del mouse sul nome del task che si vuole configurare.

2) Nella finestra `Task configuration` si può modificare il periodo di esecuzione del task.

5.4 TIPI DI DATI DERIVATI

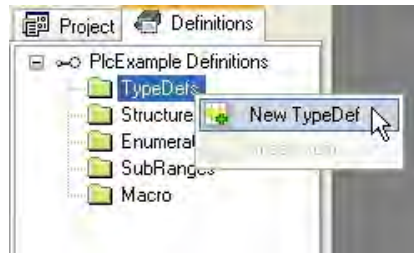
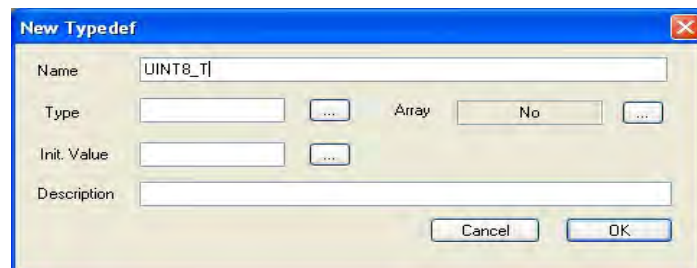
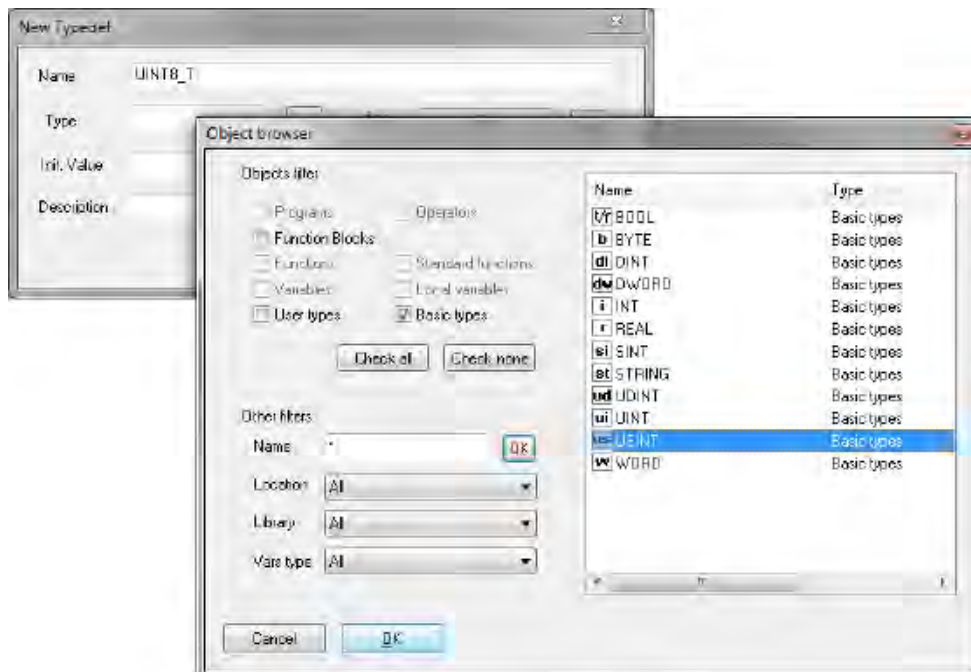
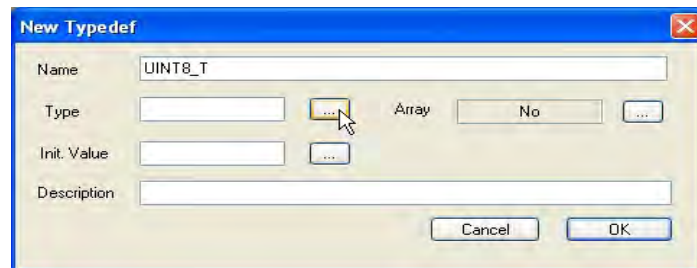
La sezione `Definitions` della finestra `Workspace` permette di definire i tipi di dati derivati.

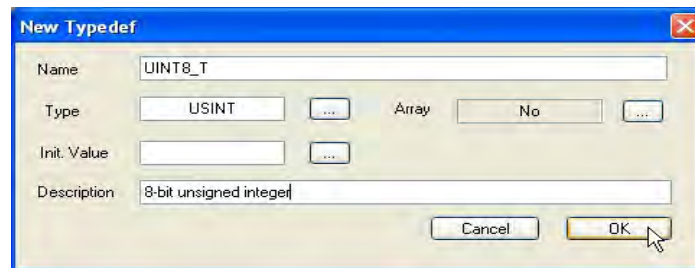
5.4.1 TYPEDEF

I paragrafi seguenti mostrano come gestire le typedef.

5.4.1.1 CREARE UN NUOVO TYPEDEF

Per creare una nuova typedef seguire la procedura seguente:

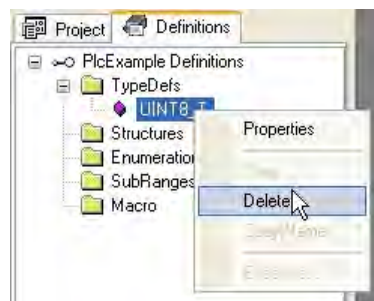
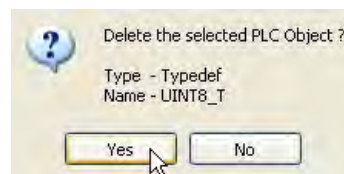
1) cliccare col tasto destro sulla cartella *Typedefs* e scegliere *New Typedef* dal menu contestuale.

2) Digitare il nome della typedef.

3) Selezionare il tipo per cui si sta definendo un alias.

(se si vuole definire un alias per un array, scegliere la dimensione dell'array).

4) Inserire una descrizione significativa (se lo si desidera) e confermare l'operazione.

5.4.1.2 MODIFICARE UN TYPEDEF

Le typedef del progetto sono elencate sotto la cartella *Typedefs*. Per modificare una typedef basta cliccare due volte sul nome.

5.4.1.3 CANCELLARE UN TYPEDEF

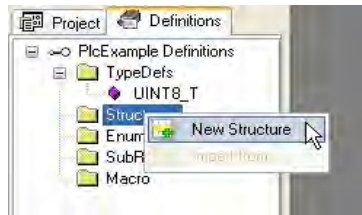
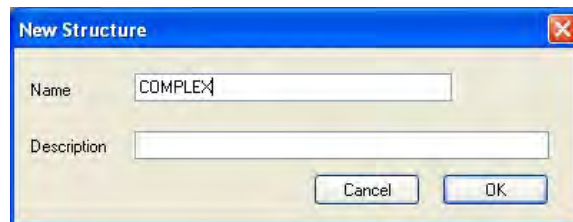
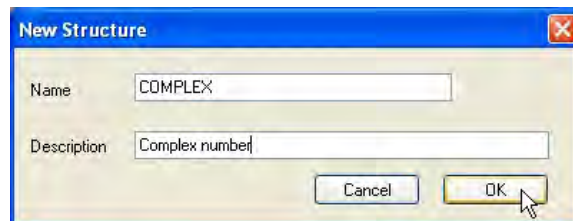
Per cancellare una typedef, seguire la seguente procedura:

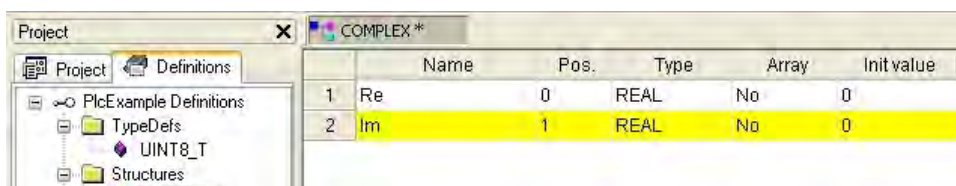
1) Cliccare col tasto destro sul nome della typedef e scegliere *Delete* dal menu contestuale.

2) Confermare la scelta.

5.4.2 STRUTTURE

I paragrafi seguenti mostrano come gestire le strutture.

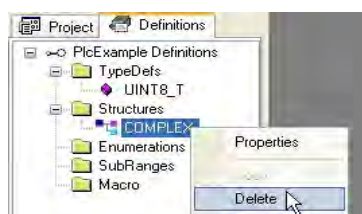
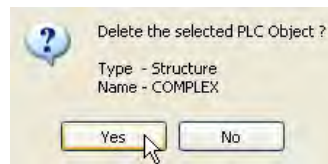
5.4.2.1 CREARE UNA NUOVA STRUTTURA

Seguire la procedura seguente per creare una nuova struttura:

1) Cliccare col tasto destro sulla cartella *Structures* e scegliere *New structure* dal menu contestuale.

2) Digitare il nome della struttura.

3) Inserire una descrizione significativa e confermare l'operazione.

5.4.2.2 MODIFICARE UNA STRUTTURA

Le strutture del progetto sono elencate sotto la cartella *Structures*. Per modificare la struttura (per esempio, definirne i campi) cliccare due volte sul suo nome.

5.4.2.3 CANCELLARE UNA STRUTTURA

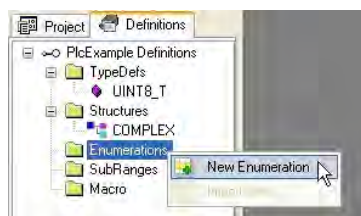
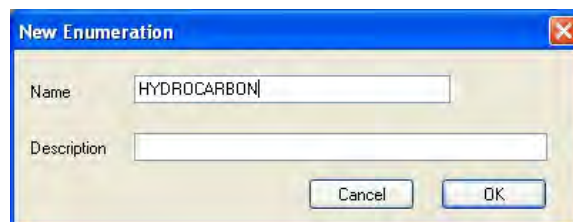
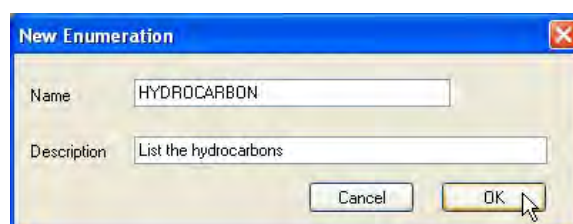
Seguire la procedura seguente per cancellare una struttura:

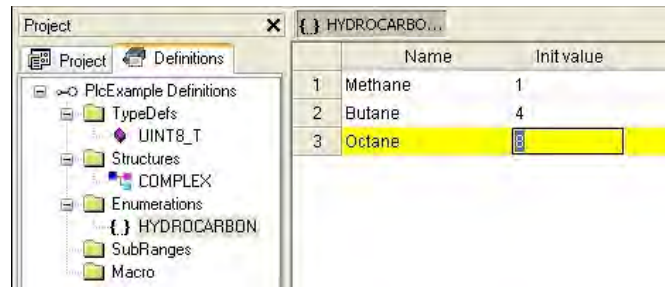
1) Cliccare col tasto destro sul nome della struttura e selezionare *Delete* dal menu contestuale.

2) Confermare la scelta.

5.4.3 ENUMERATIVI

I seguenti paragrafi mostrano come gestire gli enumerativi.

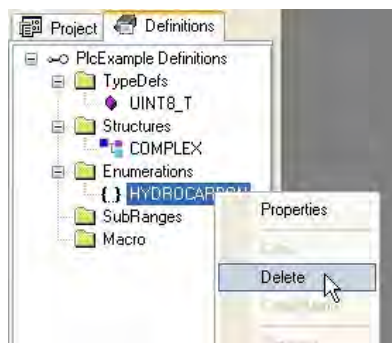
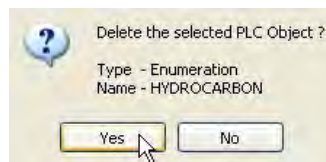
5.4.3.1 CREARE UN NUOVO ENUMERATIVO

Seguire la procedura seguente per creare un nuovo enumerativo.

1) Cliccare col tasto destro sulla cartella `Enumerations` e scegliere `New enumeration` dal menu contestuale.

2) Digitare il nome dell'enumerativo.

3) Inserire una descrizione significativa e confermare l'operazione.

5.4.3.2 MODIFICARE UN ENUMERATIVO

Gli enumerativi del progetto sono elencati sotto la cartella `Enumerations`. Per modificare un'enumerativo (per esempio, per definire i suoi valori), cliccare due volte sul suo nome.

5.4.3.3 CANCELLARE UN ENUMERATIVO

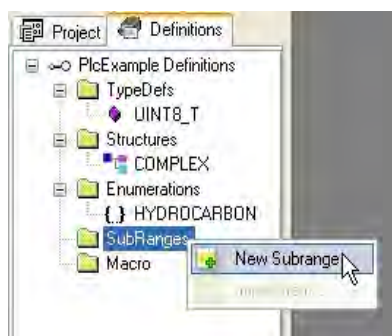
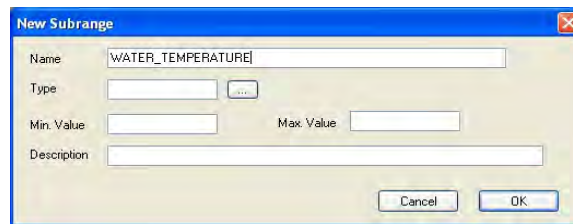
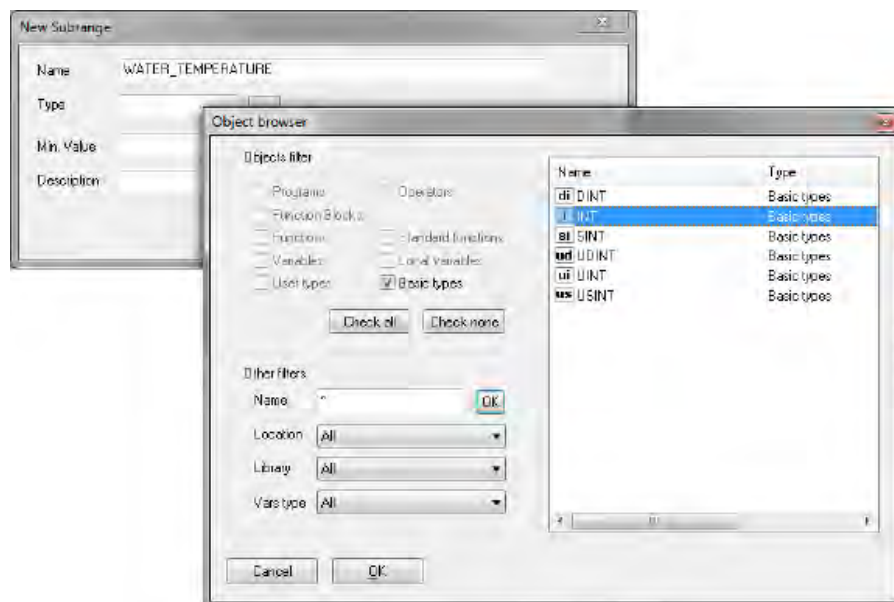
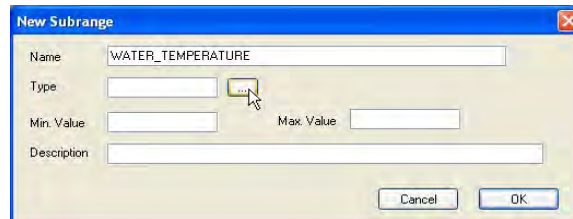
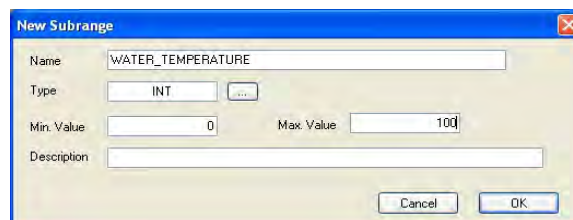
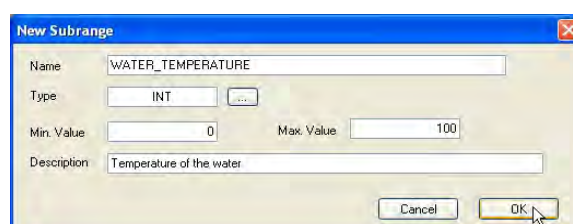
Seguire la procedura seguente per cancellare un enumerativo:

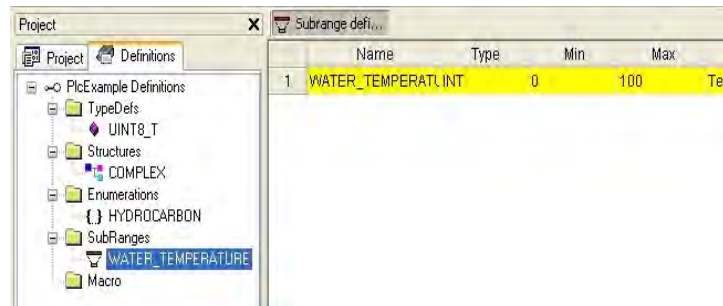
1) Cliccare col tasto destro sul nome dell'enumerativo e scegliere `Delete` dal menu contestuale.

2) Confermare la scelta.

5.4.4 SUBRANGE

Il paragrafo seguente mostra come gestire un subrange.

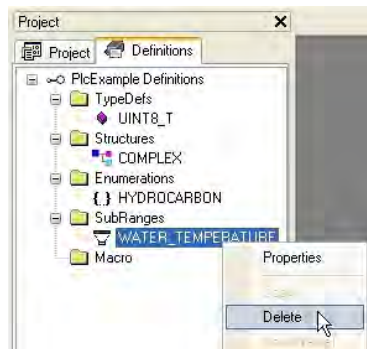
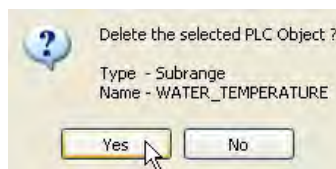
5.4.4.1 CREARE UN NUOVO SUBRANGE

Seguire la procedura seguente per creare un nuovo subrange:

1) Cliccare col tasto destro sulla cartella `Subranges` e scegliere `New subrange` dal menu contestuale.

2) Digitare il nome del subrange.

3) Scegliere il tipo di subrange.

4) Inserire un valore minimo e un valore massimo.

5) Inserire una descrizione significativa (opzionale) e confermare l'operazione.

5.4.4.2 MODIFICARE UN SUBRANGE

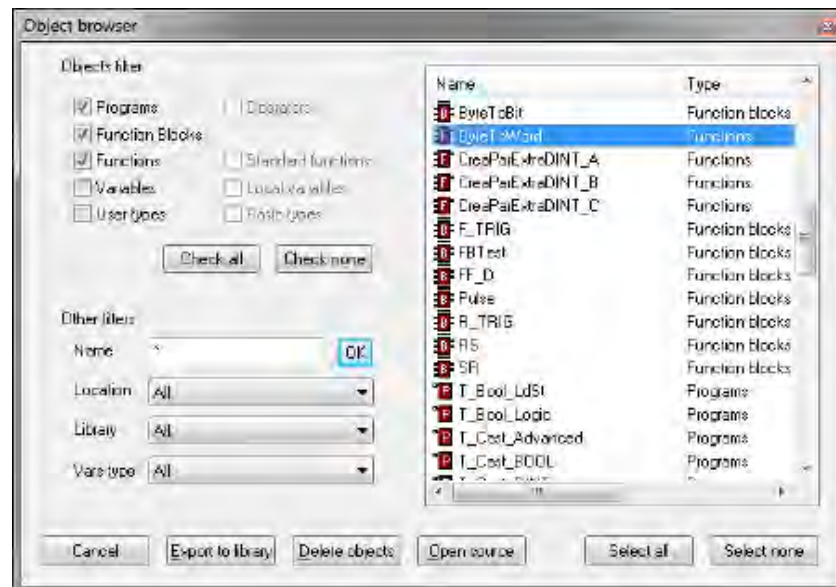
I subrange del progetto sono elencati nella cartella *Subranges*. Per modificarle basta cliccare due volte sul loro nome.

5.4.4.3 CANCELLARE UNA SOTTOCATEGORIA

Seguire la procedura seguente per cancellare un subrange:

1) Cliccare con il tasto destro sul nome del subrange e selezionare *Delete* dal menu contestuale.

2) Confermare l'operazione.

5.5 NAVIGARE IL PROGETTO

I progetti possono diventare enormi, ecco perché LogicLab fornisce due strumenti per ricercare un oggetto all'interno del progetto: la componente *Object browser* e la componente *Find in project*.

5.5.1 OBJECT BROWSER

LogicLab fornisce uno strumento manuale per navigare gli oggetti del progetto: l'*Object browser*.

Questo strumento è dipendente dal contesto, questo implica che il tipo di oggetti che possono essere selezionati e che le operazioni disponibili sugli oggetti sono diversi a seconda del contesto in cui ci si trova.

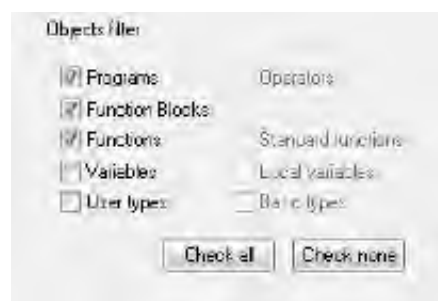
L'Object browser può essere aperto in questi tre modi:

- Modalità *Browser*.
- Modalità *Import object*.
- Modalità *Select object*.

L'interazione dell'utente con l'*Object browser* è principalmente la stessa per tutte e tre le modalità ed è descritta nel prossimo paragrafo.

5.5.1.1 CARATTERISTICHE COMUNI ED USO DELL'OBJECT BROWSER

Questa sezione descrive le caratteristiche e l'uso dell'*Object browser* che sono comuni a tutte le modalità con cui l' *Object browser* può essere utilizzato.

Objects filter

Questi sono i filtri principali dell'*Object browser*. L'utente può abilitarne (disabilitarne) uno tra i disponibili.

In questo esempio sono selezionati i filtri *Programs*, *Function Blocks* e *Functions* così oggetti di questo tipo sono riportati nella lista di oggetti.

`Variables` e `User types` possono essere selezionati dall'utente ma in questo esempio gli oggetti di questo tipo non sono mostrati nella lista di oggetti. `Operators`, `Standard functions`, `Local variables`, e `Basic types` non possono essere selezionati dall'utente (a causa del contesto).

L'utente può inoltre cliccare sul bottone `Check all` per selezionare in una volta tutti gli oggetti disponibili o può cliccare sul bottone `Check none` per deselezionare in una volta sola tutti gli oggetti.

Other filters

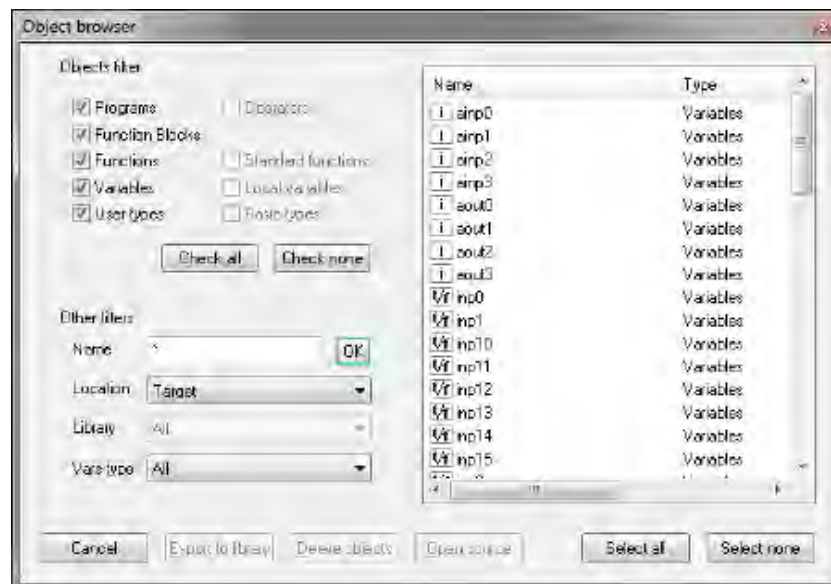
Gli oggetti selezionati possono inoltre essere filtrati per nome, posizione, libreria e tipo di variabile.

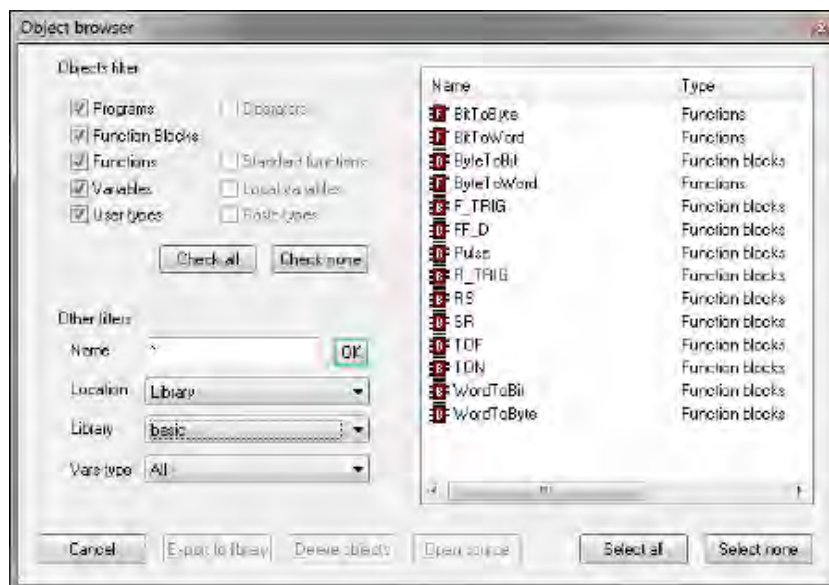
I filtri sono tutti additivi e vengono applicati immediatamente dopo l'impostazione.

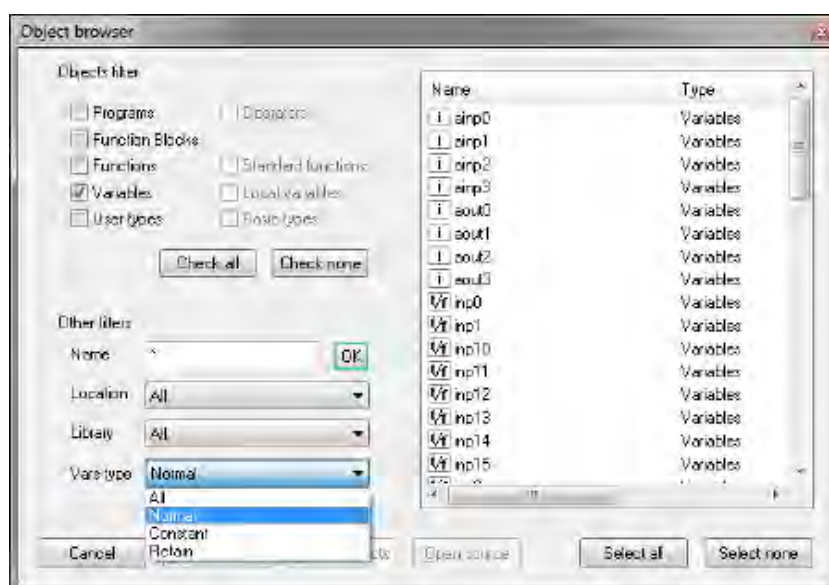
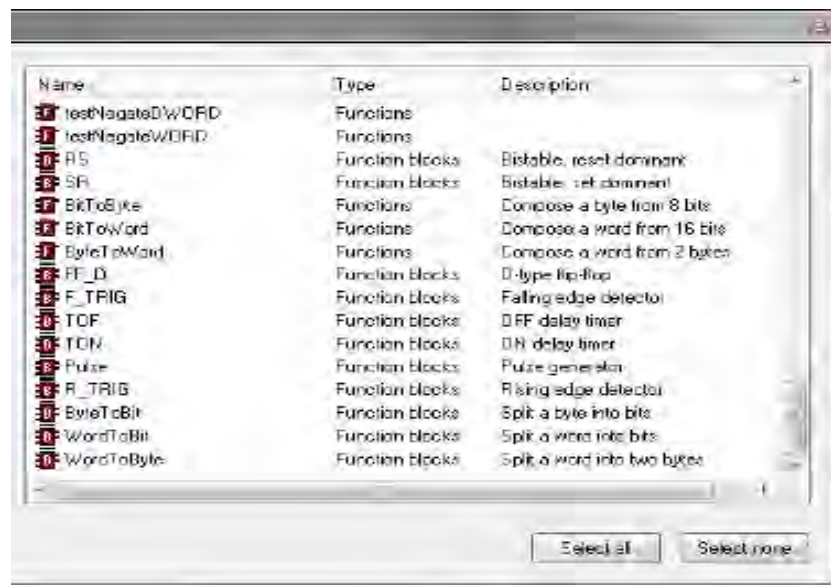
Nome

Funzione	Filtra gli oggetti in base al loro nome.
Insieme dei valori validi	Tutte le stringhe di caratteri.
Uso	Digitare una stringa per visualizzare l'oggetto specifico il cui nome corrisponde alla stringa. Usare il carattere jolly * per visualizzare tutti gli oggetti il cui nome contiene la stringa della casella di testo `Name`. Digitare * se si vuole disabilitare il filtro. Premere `Enter` quando la edit box ha il focus o premere sul bottone `OK` vicino alla edit box per applicare il filtro..
Applicabile a	Tutti i tipi di oggetto.

Posizione del simbolo

Funzione	Filtra gli oggetti in base alla loro posizione.
Insieme di valori validi	All, Project, Target, Library, Aux. Sources.
Uso	All= disabilita questo filtro. Project= oggetti dichiarati in LogicLab. Target= oggetti firmware. Library= oggetti contenuti in una libreria. In questo caso, utilizzare contemporaneamente anche il filtro `Library`, descritto in seguito. Aux sources= mostra solo fonti aux.
Applicabile a	Tutti i tipi di oggetti.

Librerie

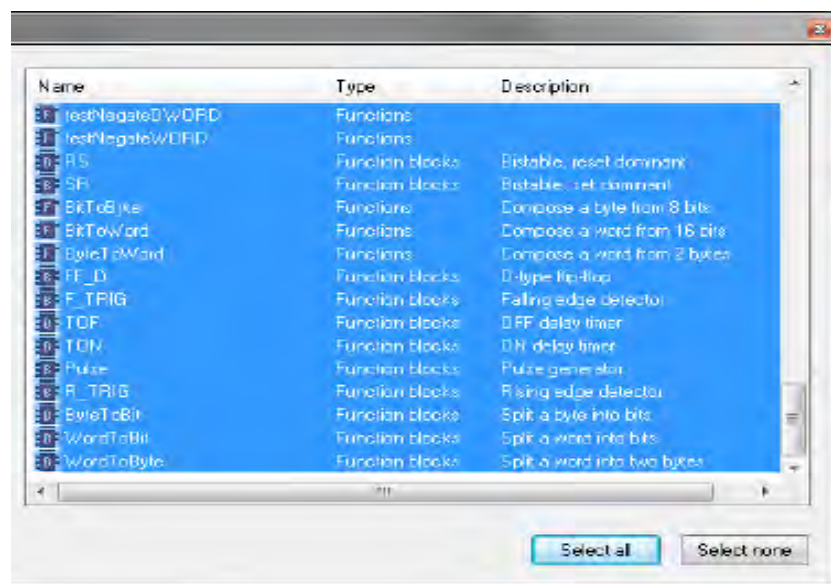
Funzioni	Completa il grado di specificazione di una ricerca sugli oggetti contenuti nelle librerie. Il valore di questo controllo è rilevante solo se il filtro `Symbol location` è impostato su `Library`.
Insieme di valori validi	All, libraryname1, libraryname2, ...
Uso	All= mostra gli oggetti contenuti in una libreria qualunque. LibrarynameN= mostra solo gli oggetti contenuti nella libreria chiamata librarynameN.
Applicabile a	Tutti i tipi di oggetto.

Tipi di variabili

Funzioni	Filtra le variabili globali e le variabili di sistema (dette anche variabili del firmware) in base al loro tipo.
Insieme di valori validi	All, Normal, Constant, Retain
Uso	All= mostra tutte le variabili globali e di sistema. Normal= mostra solo le variabili globali normali. Constant= mostra solo le costanti. Retain= mostra solo le variabili retain.
Applicabile a	Variabili.

Lista di oggetti

`Object list` mostra tutti gli oggetti filtrati. La lista può essere ordinata in maniera ascendente o discendente cliccando sull'intestazione della colonna. E' inoltre possibile ordinare gli oggetti per *Name*, *Type*, or *Description*.

Facendo doppio click su un oggetto, l'utente può eseguire le operazioni di default associate (le azioni sono le stesse delle azioni dei bottoni di *OK*, `Import object`, o `Open source`).

Quando è permessa la selezione multipla degli oggetti, sono visibili i bottoni `Select all` e `Select none`.

E' possibile selezionare tutti gli oggetti cliccando sul bottone `Select all. Select none` deseleziona tutti gli oggetti.

I pulsanti sono abilitati se almeno un oggetto è selezionato nella lista.

Ridimensionare

La finestra può essere ridimensionata, il cursore cambia lungo il bordo della finestra di dialogo e permette all'utente di ridimensionare la finestra.

Quando viene riaperta la finestra di dialogo *Object browser* tiene la stessa dimensione e posizione dell'utilizzo precedente.

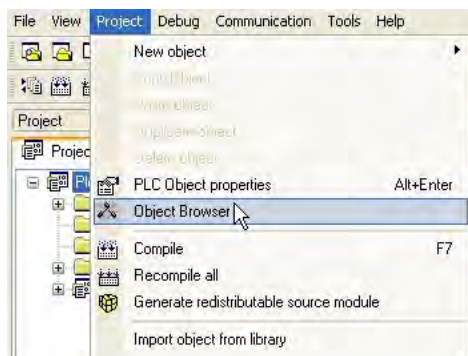
Chiudere la finestra

Per chiudere la finestra *Object browser* sono disponibili due opzioni:

- Premere il bottone vicino alla fine del bordo destro della barra del titolo.

- Premere il bottone *Cancel/OK*.

5.5.1.2 USARE UN OBJECT BROWSER COME UN BROWSER

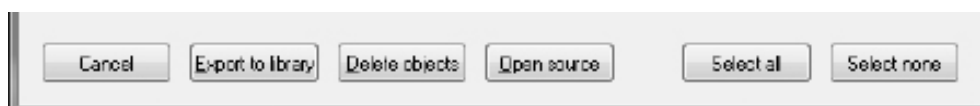
Per usare l '*Object browser* in questo modo, cliccare su *Object browser* nel menu *Project.* Questo fa apparire una finestra di dialogo che consente di navigare tra gli oggetti del progetto attualmente aperto.

Oggetti disponibili

In questa modalità si possono elencare gli oggetti di questo tipo:

- Program.
- Function Block.
- Function.
- Variables.
- User type.

Questi oggetti possono essere selezionati o deselezionati dalla sezione *Objects filter* per mostrare o nascondere gli oggetti nella lista.

Altri tipi di oggetti (Operator, Standard function, Local variables, Basic types) non possono essere navigati in questo contesto così questi sono deselezionati e non abilitati.

Operazioni disponibili

Le operazioni disponibili in questa modalità sono :

Open source, operazioni predefinite sul doppio click di un oggetto	
Function	Apre l'editor con cui è stato creato l'oggetto selezionato e visualizza il relativo codice sorgente.
Use	Se l'oggetto è un programma o una funzione o un function block, questo bottone apre il relativo editor di codice sorgente. Se l'oggetto è una variabile, questo bottone apre l'editor delle variabili. Selezionare l'oggetto il cui editor si vuole aprire, quindi premere sul bottone *Open source.*

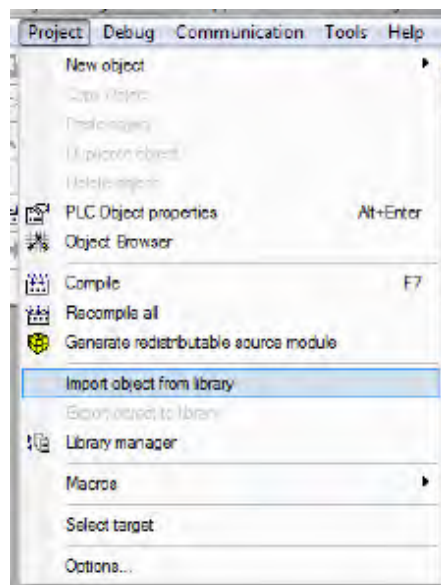
Esportare in una libreria	
Function	Per esportare un oggetto ad una libreria.
Use	Selezionare l'oggetto che si vuole esportare, quindi premere il bottone *Export to library.*

Cancellare gli oggetti	
Function	Permette di eliminare un oggetto.
Use	Selezionare l'oggetto che si vuole eliminare, quindi premere il bottone *Delete object*.

Selezione multipla

La selezione multipla è permessa per queste modalità, sono visibili i bottoni *Select all* e *Select none.*

5.5.1.3 USARE L'OBJECT BROWSER PER IMPORTARE

L' *Object browser* è anche usato per supportare l'importazione di oggetti nel progetto da una libreria esterna desiderata. Selezionare *Import object from library* dal menu *Project,* quindi scegliere la libreria desiderata.

Oggetti disponibili

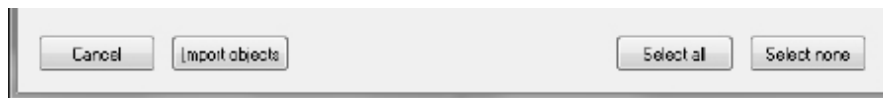
In questa modalità

In questa modalità si possono elencare gli oggetti di questo tipo:

- Program.
- Function block.
- Function.
- Variables.
- User types.

Questi oggetti possono essere selezionati o deselezionati nella sezione *Objects filter* per mostrare o nascondere gli oggetti nella lista.

Altri tipi di oggetti (Operator, Standard functions, Local variables, Basic types) non possono essere importati quindi sono disabilitati e deselezionati.

Operazioni disponibili

L'unica operazione supportata in questa modalità è *Import objects.* E' possibile importare gli oggetti selezionati facendo clic sul pulsante *Import objects* o facendo doppio clic su uno degli oggetti della lista.

Selezione multipla

In questa modalità è disponibile la selezione multipla, sono visibili i bottoni S*elect all* e *Select none.*

5.5.1.4 USARE L'OBJECT BROWSER PER L'OGGETTO SELECTION

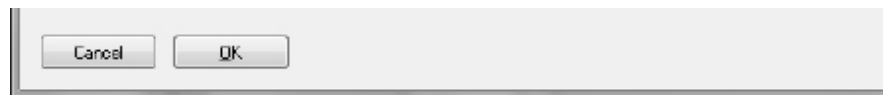
La finestra Object browser è utilizzabile per molte operazioni che richiedono la selezione di un singolo oggetto PLC.

Così l'Object browser può essere usato per selezionare il programma da aggiungere ad un task, per selezionare il tipo di una variabile, per selezionare un oggetto, per selezionare un oggetto da cercare nel progetto, etc...

Oggetti disponibili

Gli oggetti disponibili sono strettamente dipendenti dal contesto, per esempio nell'operazione di assegnamento di un programma ad un task i soli oggetti disponibili sono gli oggetti dei programmi.

E' possibile che non tutti gli oggetti disponibili siano selezionati in modo predefinito.

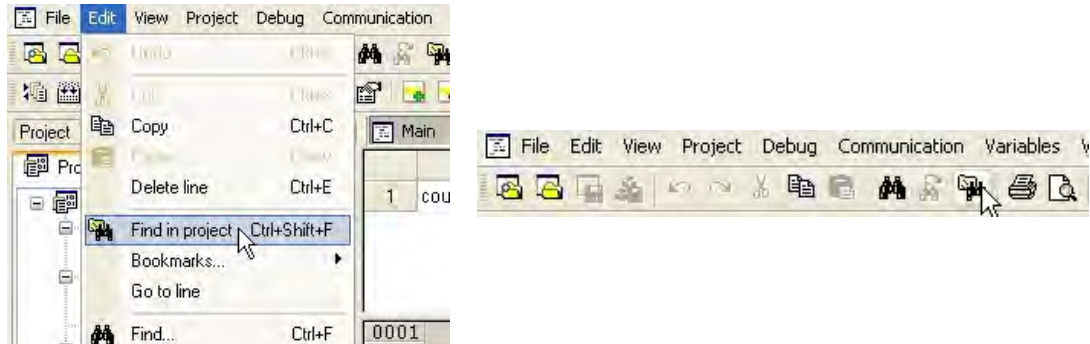
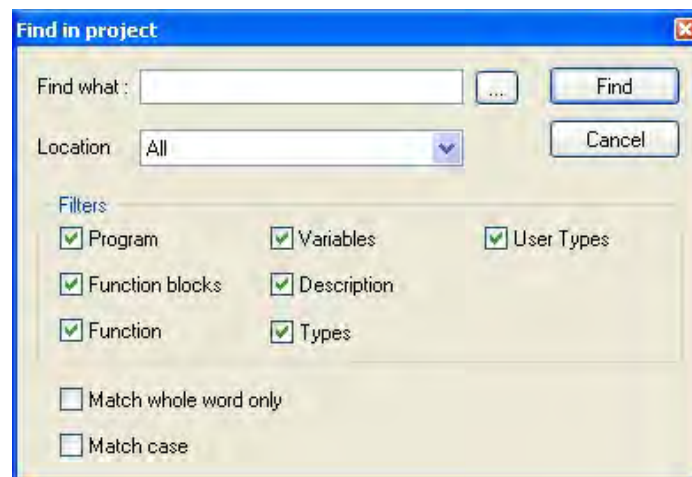
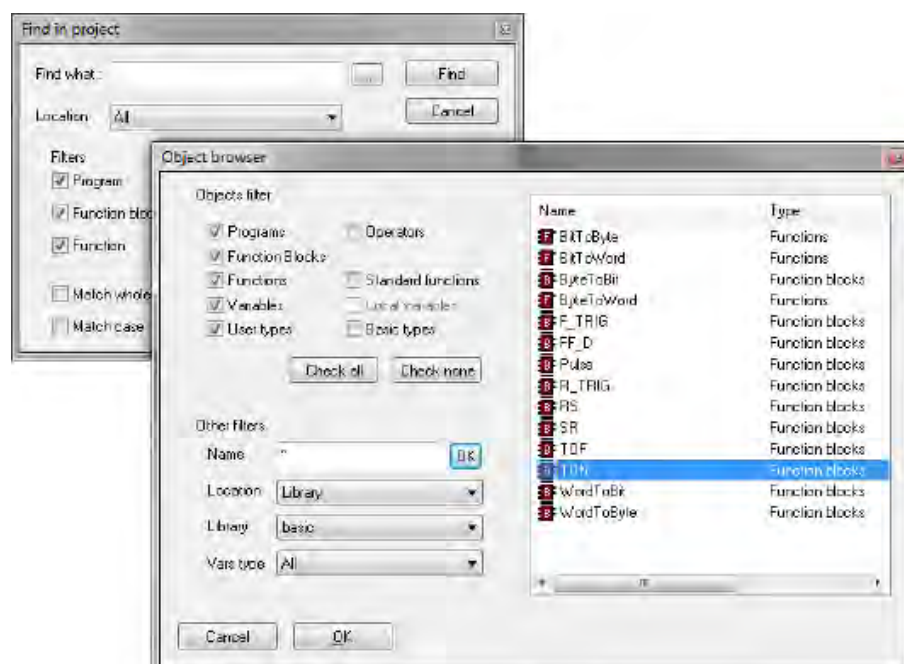
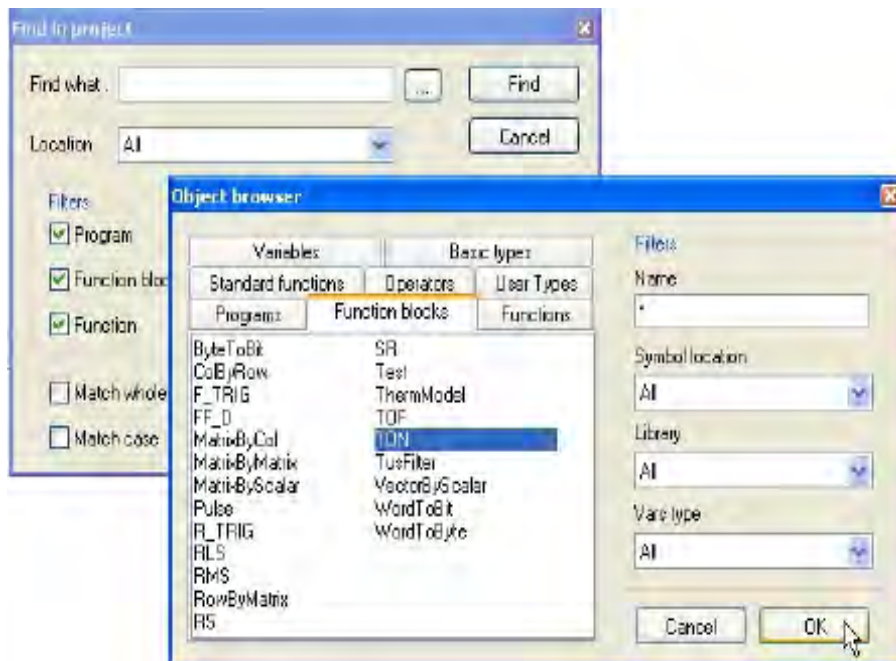
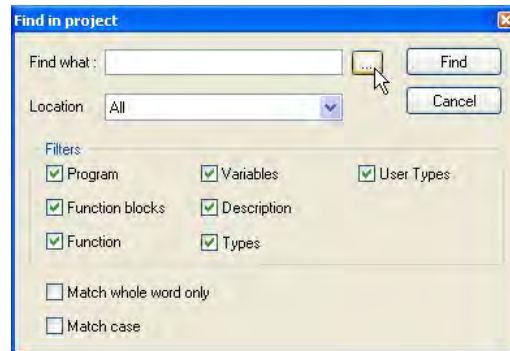
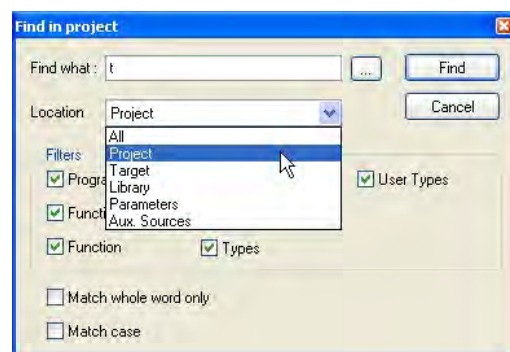
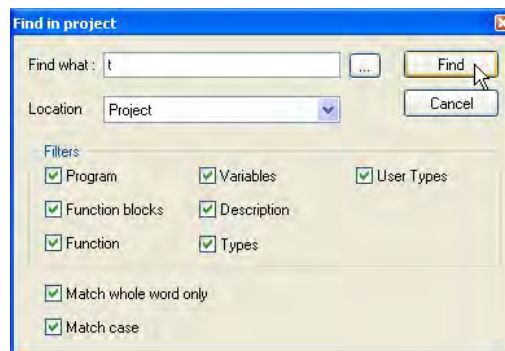
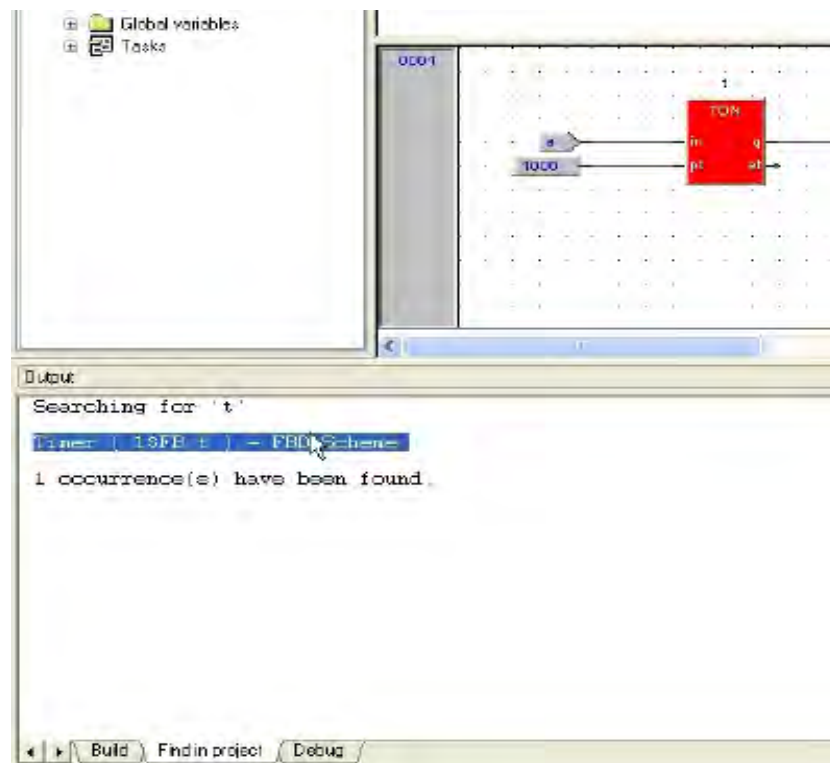
Operazioni disponibili

In questo modo è possibile selezionare un singolo oggetto facendo doppio click sulla lista o cliccando sul bottone *OK,* quindi la finestra viene automaticamente chiusa.

Selezione multipla

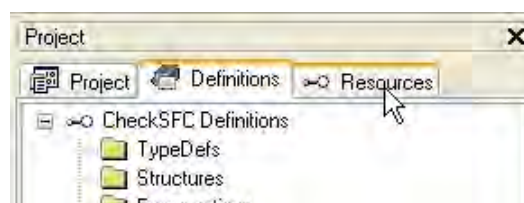
La selezione multipla per questa modalità non è disponibile. I bottoni *Select all* e *Select none* non sono visibili.

5.5.2 RICERCA CON IL COMANDO FIND IN PROJECT

Il comando *Find in project* recupera tutte le ricorrenze di una specifica stringa di carattere nel progetto. Seguire la procedura sottostante per utilizzare lo strumento correttamente.

1) Cliccare *Find in project.*. nel menu *Edit* o nella barra *Main*.

Ciò farà apparire la seguente finestra di dialogo.

2) Nella casella di testo *Find what*, digitare il nome dell'oggetto che si vuole cercare.

Altrimenti, cliccare il tasto *Browse* alla destra della casella di testo, e selezionare il nome dell'oggetto dalla lista di tutte le voci esistenti.

3) Selezionare uno dei valori elencati nella casella di selezione *Location*, per specificare una restrizione sulla posizione dell'oggetto da ricercare.

4) La cornice chiamata *Filters* contiene sette caselline, ciascuna delle quali, se evidenziata, permette la ricerca della stringa a cui l'oggetto si riferisce.
5) Spuntare *Match whole word only* per ricercare solo la parola per intero.
6) Spuntare *Match case* per effettuare la ricerca facendo distinzione tra Maiuscole e Minuscole.
7) Premere *Find* per avviare la ricerca, altrimenti cliccare su *Cancel* per abbandonare.

I risultati verranno riportati nella casella `Find in project` della finestra `Output`.

5.6 LAVORARE CON LE ESTENSIONI DI LOGICLAB

La finestra Workspace di LogicLab potrebbe includere una sezione i cui contenuti dipendono completamente dal target device con cui IDE sta interagendo: il pannello `Resources`.

Se il pannello `Resources` è visibile, si può avere accesso ad alcune componenti aggiuntive relative al target device (elementi di configurazione, schemi, wizard ,e così via).

Le informazioni relative a questi componenti possono essere trovate in un documento separato: fare riferimento al fornitore dell'hardware per i dettagli.

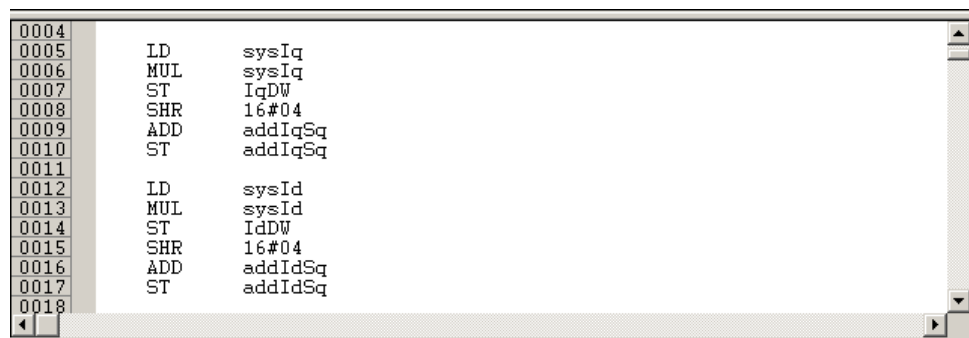
6. MODIFICARE IL CODICE SORGENTE

Editor PLC

LogicLab comprende cinque editor di codice sorgente, che supportano l'intera gamma di linguaggi di programmazione IEC 61131-3: Instruction List (IL), Structured Text (ST), Ladder Diagram (LD), Function Block Diagram (FBD) e Sequential Function Chart (SFC).

In aggiunta, LogicLab comprende un editor a griglia per fornire supporto all'utente nella definizione delle variabili.

Questo capitolo è dedicato a tutti questi editor.

6.1 EDITOR INSTRUCTION LIST (IL)

L'editor IL permette di codificare e modificare le POU utilizzando IL (Instruction List), uno dei linguaggi ammessi in IEC.

6.1.1 MODIFICARE LE FUNZIONI

L'editor IL è fornito di funzioni comuni alla maggior parte degli editor funzionanti su una piattaforma Windows, ovvero:

- selezione del testo.
- Operatori *Cut*, *Copy* e *Paste*.
- Funzioni *Find and Replace*.
- Drag-and-drop del testo selezionato.

La maggior parte di queste funzioni è accessibile dal menu *Edit* o dalla barra *Main*.

6.1.2 RIFERIMENTO AGLI OGGETTI PLC

Se c'è la necessità di aggiungere al codice IL un riferimento ad un oggetto PLC esistente, sono possibili due opzioni::

- Digitare direttamente il nome dell'oggetto PLC.
- Trascinarlo in una posizione adatta. Per esempio, le variabili globali possono essere prese dalla finestra *Workspace*, mentre gli operatori standard e le funzioni incorporate possono essere trascinate dalla finestra *Library*, mentre le variabili locali possono essere selezionate dall'editor delle variabili locali.

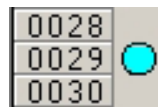
6.1.3 LOCAZIONE AUTOMATICA DELL'ERRORE

L'editor IL mostra automaticamente anche la posizione degli errori del compilatore. Per sapere dove si è verificato un errore di questo tipo, cliccare due volte sulla linea di errore corrispondente sulla barra *Output*.

6.1.4 SEGNALIBRI

E' possibile impostare i segnalibri per segnalare la riga maggiormente frequentato del file sorgente. Una volta che il segnalibro è posizionato, lo si può raggiungere con un semplice comando della tastiera. E' inoltre possibile rimuoverlo una volta che non sia più necessario.

6.1.4.1 IMPOSTARE UN SEGNALIBRO

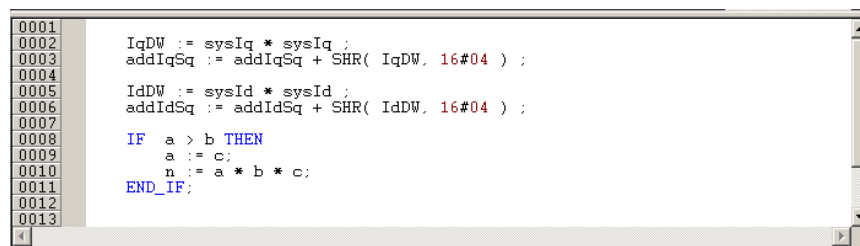
Spostare il punto di inserzione sulla linea in cui si vuole posizionare il segnalibro, poi premere *CTRL+F2*. La linea sarà evidenziata a margine con un cerchio azzurro.

6.1.4.2 SALTARE AD UN SEGNALIBRO

Premere ripetutamente *F2*, fino al raggiungimento della linea desiderata.

6.1.4.3 RIMUOVERE UN SEGNALIBRO

Spostare il cursore in un punto qualunque sulla linea contenente il segnalibro, poi premere *CTRL+F2*.

6.2 EDITOR STRUCTURED TEXT (ST)

```
0001
0002    IqDW  := sysIq * sysIq ;
0003    addIqSq := addIqSq + SHR( IqDW, 16#04 ) ;
0004
0005    IdDW  := sysId * sysId ;
0006    addIdSq := addIdSq + SHR( IdDW, 16#04 ) ;
0007
0008    IF  a > b THEN
0009        a := c;
0010        n := a * b * c;
0011    END_IF;
0012
0013
```

L'editor ST permette di codificare e modificare le POU utilizzando ST (Structured Text), uno dei linguaggi ammessi in IEC.

6.2.1 CREARE E MODIFICARE OGGETTI ST

Vedere la sezione Creare e modificare POU (paragrafi 5.1.1 e 5.1.2).

6.2.2 MODIFICARE LE FUNZIONI

L'editor ST è fornito di funzioni comuni alla maggior parte degli editor funzionanti su una piattaforma Windows, ovvero:

- Selezione del testo.
- Operatori `Cut`, `Copy`, e `Paste`.
- Funzioni `Find` e `Replace`.
- Drag-and-drop del testo selezionato.

La maggior parte di queste funzioni è accessibile dal menu `Edit` o dalla barra `Main`.

6.2.3 RIFERIMENTO AD OGGETTI PLC

Se c'è la necessità di aggiungere al codice ST un riferimento ad un oggetto PLC esistente, sono possibili due opzioni:

- Digitare direttamente il nome dell'oggetto PLC.
- Trascinarlo in una posizione adatta. Per esempio, le variabili globali possono essere prese dalla finestra `Workspace`, mentre gli operatori standard e le funzioni incorporate possono essere trascinate dalla finestra `Library`, mentre le variabili locali possono essere selezionate dall'editor delle variabili locali.

6.2.4 LOCAZIONE AUTOMATICA DELL'ERRORE

L'editor ST mostra automaticamente anche la posizione degli errori del compilatore. Per sapere dove si è verificato un errore di questo tipo cliccare due volte la linea di errore corrispondente sulla barra `Output`.

6.2.5 SEGNALIBRI

E' possibile impostare i segnalibri per segnalare le righe maggiormente frequentate del file sorgente. Una volta che il segnalibro è posizionato, lo si può raggiungere con un semplice comando della tastiera. E' inoltre possibile rimuoverlo una volta che non sia più necessario.

6.2.5.1 IMPOSTARE UN SEGNALIBRO

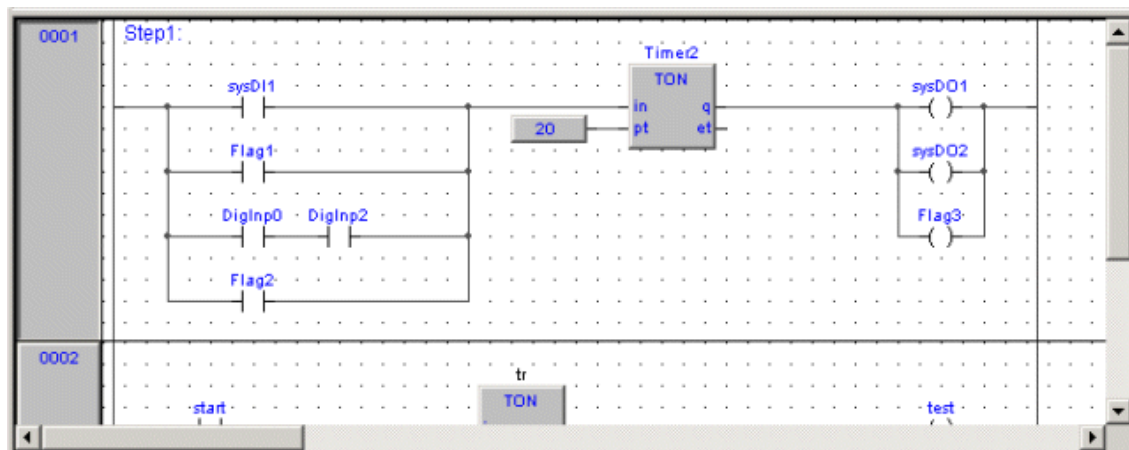
Spostare il punto di inserzione sulla linea in cui si vuole posizionare il segnalibro, poi premere `CTRL+F2`. La linea sarà evidenziata a margine con un cerchio azzurro.

6.2.5.2 SALTARE AD UN SEGNALIBRO

Premere ripetutamente `F2`, fino al raggiungimento della linea desiderata.

6.2.5.3 RIMUOVERE UN SEGNALIBRO

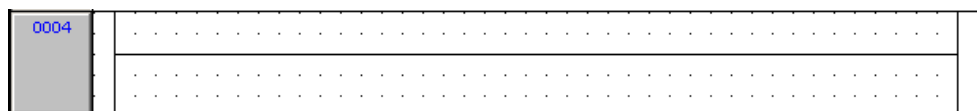
Spostare il cursore in un punto qualunque sulla linea contenente il segnalibro, poi premere `CTRL+F2`.

6.3 EDITOR LADDER DIAGRAM (LD)

L'editor LD permette di codificare e modificare le POU utilizzando LD (Ladder Diagram), uno dei linguaggi ammessi in IEC.

6.3.1 CREARE UN NUOVO DOCUMENTO LD

Vedere la sezione Creare e modificare POU (paragrafi 5.1.1 e 5.1.2).

6.3.2 AGGIUNGERE/RIMUOVERE NETWORK

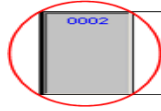
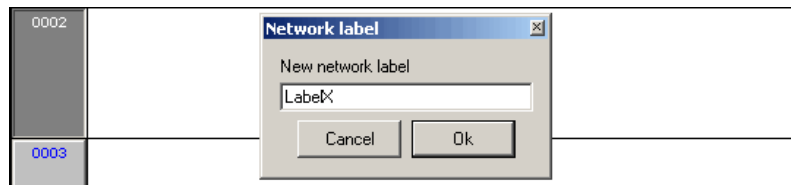
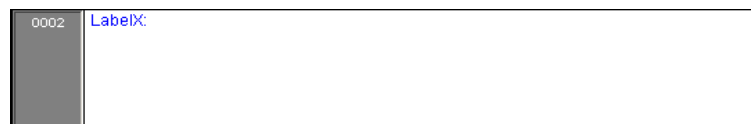
Ogni POU codificata in LD consiste in una sequenza di network. Un network è definito come un set massimo di elementi grafici interconnessi. Le estremità alta e bassa di ogni network sono fissate da due linee dritte, mentre ogni network è delimitato a sinistra da un tasto grigio in rilievo che contiene il numero del network.

Su ogni network LD sono rappresentati i power rails di destra e di sinistra, secondo le indicazione del linguaggio LD.

Sul nuovo LD network una linea orizzontale unisce i due power rails. E' definita "power link". Su questo link devono essere sistemati tutti gli elementi LD (contatti, uscite e blocchi).

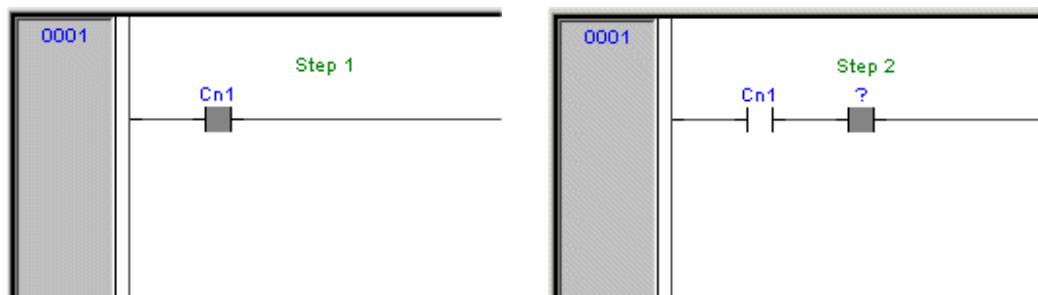
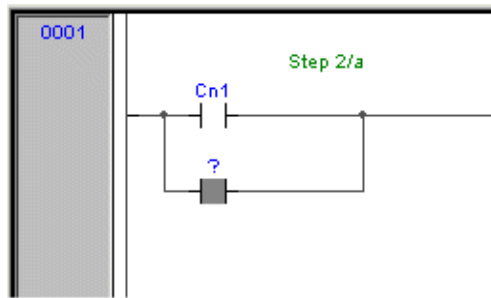
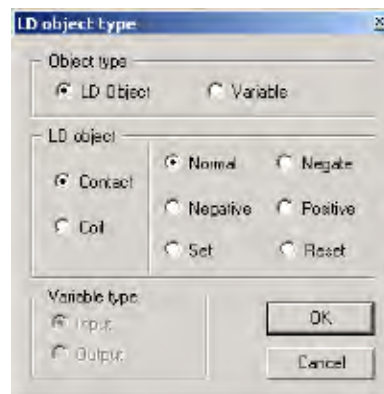
Sul network si possono eseguire le seguenti operazioni:

- Per aggiungere un nuovo network, cliccare *Network>New* nel menu *Scheme*, o premere uno dei tasti equivalenti sulla barra *Network*.
- Per assegnare un'etichetta ad un network selezionato, eseguire il comando *Network>Label* del menu *Scheme*. Questo renderà possibile raggiungere direttamente il network etichettato.
- Per visualizzare una griglia di sfondo che aiuti ad allineare gli oggetti, premere *View grid* sulla barra *Network*.
- Per aggiungere un commento, premere il tasto *Command* nella barra *FBD*.

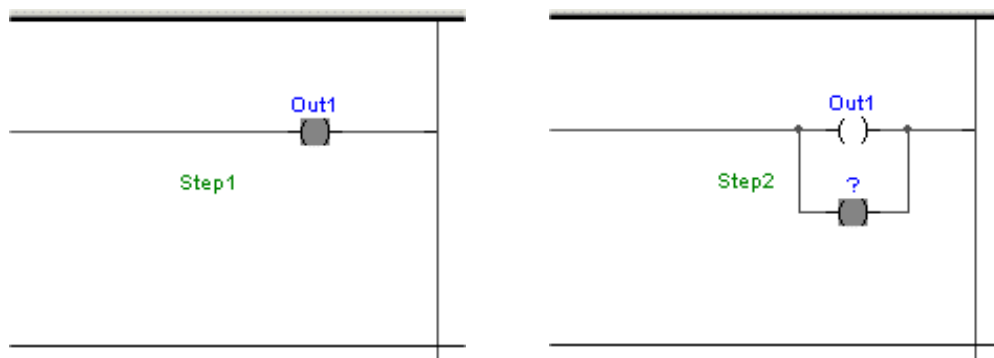
6.3.3 ETICHETTARE I NETWORK

Si può modificare l'ordine di esecuzione usuale dei network attraverso un' istruzione jump, che trasferisce il controllo del programma ad un network etichettato. Per assegnare un'etichetta ad un network, cliccare due volte il tasto grigio in rilievo che porta il numero del network, a sinistra.

Ciò farà apparire una finestra di dialogo, dove sarà possibile digitare l'etichetta che si vuole associare al network selezionato..

Premendo OK, l'etichetta sarà stampata nell'angolo in alto a sinistra del network selezionato.

6.3.4 INSERIRE I CONTATTI

Per inserire nuovi contatti sul network seguire una di queste opzioni:

- Selezionare un contatto, un blocco o una connessione. Selezionare il modo di inserimento tra seriale o parallelo (usando il tasto della barra *LD* o il menu *Scheme*). Inserire il contatto appropriato (usando il tasto *LD* della barra, *Scheme>Object>New* o l'opzione nel menu di pop-up). Per l'inserimento seriale, il nuovo contatto verrà inserito nella parte destra del contatto/blocco selezionato o nel mezzo della connessione selezionata a seconda di quale sia l'elemento selezionato prima dell'inserimento. Per l'inserimento parallelo, possono essere selezionati più contatti/ blocchi prima dell'inserimento. Il nuovo contatto verrà in questo caso posizionato alla fine del blocco di selezione.

- Trascinare una variabile booleana nella posizione desiderata sopra ad una connessione. Per esempio, le variabili globali possono essere prese dalla finestra Workspace, mentre le variabili locali dall'editor di variabili locali. Apparirà la finestra di dialogo mostrata qui sotto, e richiederà di definire se la variabile debba essere inserita come contatto, uscita o variabile (come gli schemi FBD). Scegliere il tipo di contatto appropriato. I contatti inseriti tramite il trascinamento sono sempre inseriti in serie.

6.3.5 INSERIRE LE USCITE

Per inserire nuove uscite su un network procedere come segue:

- Premere uno dei tasti di uscita nella barra *LD*. La nuova uscita sarà inserita e collegata al power rail di destra. Se sono già presenti delle uscite nel network, la nuova uscita sarà aggiunta parallelamente rispetto alle precedenti.

- Trascinare una variabile booleana sul network. Per esempio, le variabili globali possono essere prese dalla finestra *Workspace*, mentre le variabili locali dall'editor di variabili locali. Apparirà la finestra di dialogo mostrata qui sotto, e richiederà di definire se la variabile debba essere inserita come contatto, uscita o variabile. Scegliere il tipo di uscita appropriato.

6.3.6 INSERIRE I BLOCCHI

Gli operatori, le funzioni e i blocchi funzione possono essere inseriti in un network *LD* nei modi seguenti:

- Sul power link, come contatti e uscite.
- Fuori dal power link (per fare ciò, seguire le indicazioni dei blocchi FBD).

Per inserire i blocchi sul network seguire una delle seguenti opzioni:

- Selezionare un contatto, una connessione o un blocco e cliccare *Object>New* nel menu *Scheme*.
- Selezionare un contatto, una connessione o un blocco e premere il tasto *New block* nella barra *FBD*. Ciò farà comparire una finestra di dialogo che elenca tutti gli oggetti del progetto; scegliere una delle voci dell'elenco. Se il blocco è una costante, un'istruzione return o un' istruzione jump, è possibile premere direttamente i tasti corrispondenti sulla barra *FBD*.
- Trascinare l'oggetto selezionato (dalla finestra *Workspace*, dalla finestra *Libraries* o dall'editor variabili locali) sulla connessione desiderata.

I pins in alto saranno connessi al power link. I pins EN/ENO dovrebbero essere attivati prima dell'inserimento.

6.3.7 MODIFICARE LE PROPRIETA' DELLE USCITE E DEI CONTATTI

Il tipo di contatto (normale, negato) o di uscita (normale,negata, sempre accesa, sempre spenta) può essere cambiato attraverso una delle seguenti operazioni:

- Doppio click sull'elemento (contatto o uscita).
- Selezionare l'elemento e poi premere *Enter*.
- Selezionare l'elemento, attivare il menu di pop-up col tasto destro del mouse, poi selezionare *Properties*.

Apparirà un'apposita finestra di dialogo. Selezionare il tipo di elemento desiderato della lista presentata e premere *OK*.

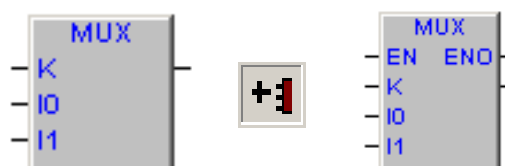
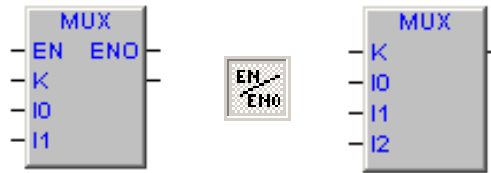
6.3.8 MODIFICARE I NETWORK

L'editor LD è fornito di funzioni comuni alla maggior parte di applicazione grafiche funzionanti sulla piattaforma Windows, ovvero:

- Selezione di un blocco.
- Selezione di un set di blocchi premendo il tasto *Shift+Right* e disegnando una cornice che includa i blocchi da selezionare.
- Operazioni di *Cut*, *Copy* e *Paste* sia di un singolo blocco sia di un set di blocchi.
- Drag-and-drop.

Tutte le funzioni nominate sono accessibili tramite il menu *Edit* o la barra *Main*.

6.3.9 MODIFICARE LE PROPRIETA' DEI BLOCCHI

- Cliccare *Increment pins +* nel menu *Scheme*, o premere il tasto *Inc pins* nella barra *FBD*, per aumentare il numero di pins di input di alcuni operatori e funzioni incorporate.

- Cliccare `Enable EN/ENO` pins nel menu `Scheme`, o premere il tasto `EN/ENO` nella barra `FBD`, per visualizzare i pins input e output abilitati.

- Cliccare `Object. Instance name` nel menu `Scheme`, o premere il tasto `FBD properties` nella barra `FBD`, per cambiare il nome dell' istanza di un blocco funzione.

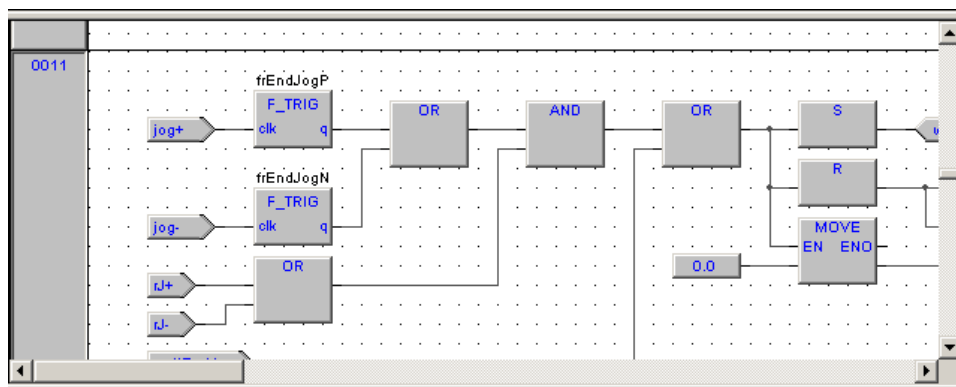
6.3.10 ACQUISIRE INFORMAZIONI DA UN BLOCCO

E' sempre possibile informarsi su un blocco aggiunto ad un documento LD selezionandolo e compiendo una delle seguenti operazioni:

- Cliccare `Object>Open source` nel menu `Scheme`, o premere il tasto `View source` nella barra `FBD`, per aprire il codice sorgente di un blocco.
- Cliccare `Object properties` nel menu `Scheme`, o premere il tasto `FBD properties` nella barra `FBD`, per vedere le proprietà e i pins di input e output del blocco selezionato.

6.3.11 RILEVAZIONE AUTOMATICA DELL'ERRORE

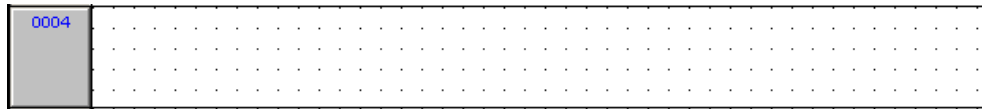
L'editor LD mostra automaticamente anche la posizione degli errori del compilatore. Per raggiungere il blocco dove l'errore ha avuto luogo, cliccare due volte sulla linea corrispondente dell'errore nella barra `Output`.

6.4 EDITOR FUNCTION BLOCK DIAGRAM (FBD)

L'editor FBD permette di codificare e modificare le POU usando FBD (Function Block Diagram), uno dei linguaggi ammessi in IEC.

6.4.1 CREARE UN NUOVO DOCUMENTO FBD

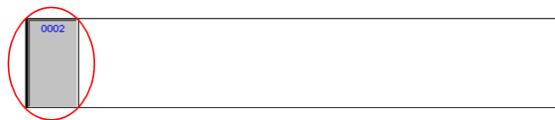
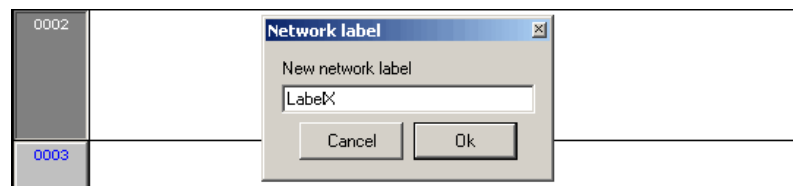
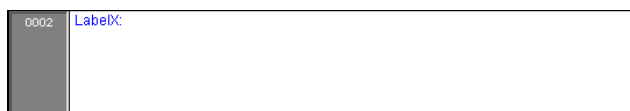
Vedere la sezione "Creare e modificare le POU" (paragrafi 5.1.1 e 5.1.2).

6.4.2 AGGIUNGERE/RIMUOVERE NETWORK

Ogni POU codificata in FBD è composta da una sequenza di network. Un network è definito come un insieme massimo di elementi grafici interconnessi. Le estremità alta e bassa di ogni network sono fissate da due linee dritte, mentre a sinistra ogni network è delimitato da un tasto grigio in rilievo che contiene il numero del network.

Sul network si possono eseguire le seguenti operazioni:

- Per aggiungere un nuovo network, cliccare *Network>New* nel menu *Scheme*, o premere uno dei tasti equivalenti nella barra *Network*.
- Per assegnare un'etichetta ad un network selezionato, eseguire il comando *Network>Label* del menu *Scheme*. Questo renderà possibile raggiungere direttamente il network etichettato.
- Per visualizzare una griglia di sfondo che aiuti ad allineare gli oggetti, premere *View grid* sulla barra *Network*.
- Per aggiungere un commento, premere *Comment* sulla barra FBD.

6.4.3 ETICHETTARE I NETWORK

Si può modificare l'ordine di esecuzione usuale dei network attraverso un'istruzione jump, che trasferisce il controllo del programma ad un network etichettato. Per assegnare un'etichetta ad un network, cliccare due volte il tasto grigio in rilievo che porta il numero del network, sulla sinistra.

Ciò farà apparire una finestra di dialogo, dove sarà possibile digitare l'etichetta che si vuole associare al network selezionato.

Premendo OK, l'etichetta sarà stampata nell'angolo in alto a sinistra del network selezionato.

6.4.4 INSERIRE E CONNETTERE I BLOCCHI

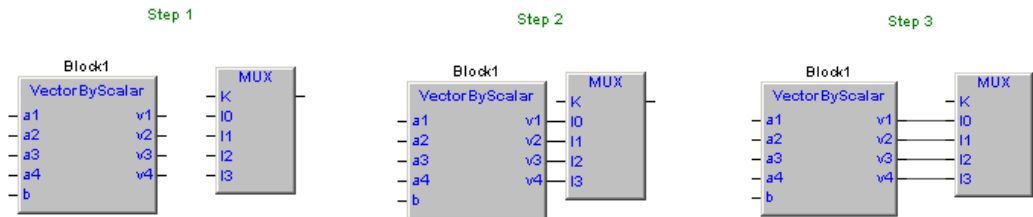
Questo paragrafo mostra come costruire un network.

Aggiungere un blocco al network vuoto, con una delle seguenti operazioni:

- Cliccare *Object>New* nel menu *Scheme*.
- Premere sul tasto *New block* nella barra FBD, ciò farà comparire una finestra di dialogo che elenca tutti gli oggetti del progetto, scegliere poi una voce dalla lista. Se il blocco è una costante, un return statement o un jump statement si può semplicemente cliccare sui tasti corrispondenti nella barra *FBD*.
- Trascinare l'oggetto selezionato nella posizione desiderata. Per esempio, le variabili globali possono essere prese dalla finestra *Workspace*, mentre gli operatori standard e le funzioni incorporate possono essere trascinate dalla finestra *Libraries*, mentre le variabili locali possono essere selezionate dall'editor delle variabili locali.

Ripetere fino a che non si sono aggiunti tutti i blocchi che costruiranno il network.

A questo punto, connettere i blocchi:

- Cliccare *Connection mode* nel menu *Edit*, o premere il tasto *Connection* della barra *FBD*, o premere semplicemente la barra spaziatrice della tastiera. Cliccare una volta il pin sorgente, poi spostare il cursore sul pin di destinazione: l'editor FBD crea una connessione tra i due.
- Se si vogliono connettere due blocchi che hanno una corrispondenza di pins uno-a-uno, si può abilitare la modalità di autoconnessione cliccando *Autoconnect* nel menu *Scheme* o premendo il tasto *Autoconnect* nella barra *Network*. Prendere poi i due blocchi e trascinarli l'uno vicino all'altro cosicché i pins corrispondenti coincidano. L'editor FBD traccia automaticamente le connessioni.

Cancellando un blocco, le sue connessioni non saranno rimosse automaticamente, ma diventeranno non valide ed evidenziate in rosso. Cliccare *Delete invalid connection* nel menu *Scheme* o digitare *CTRL+B* sulla tastiera.

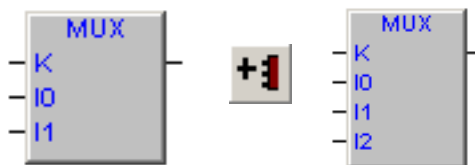
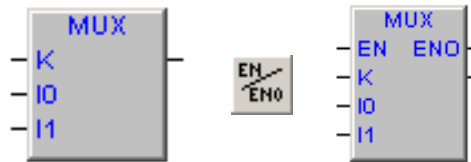
6.4.5 MODIFICARE I NETWORK

L'editor FBD è fornito di funzioni comuni alla maggior parte delle applicazione grafiche funzionanti sulla piattaforma Windows, ovvero:

- selezione di un blocco.
- Selezione di un set di blocchi premendo il tasto *Shift* + left e disegnare una cornice che include i blocchi da selezionare.
- Operazioni di *Cut*, *Copy* e *Paste* di un singolo blocco o di un set ti blocchi.
- Drag-and-drop.

Tutte le funzioni nominate sono accessibili tramite il menu *Edit* o la barra *Main*.

6.4.6 MODIFICARE LE PROPRIETA' DEI BLOCCHI

- Cliccare *Increment pins* + nel menu *Scheme*, o premere il tasto *Inc pins* nella barra *FBD*, per incrementare il numero di pins di input di alcuni operatori e funzioni incorporate.

- Cliccare *Enable EN/ENO pins* nel menu *Scheme*, o premere il tasto *EN/ENO* nella barra FBD, per visualizzare le pins input e output abilitate.

- Cliccare *Object>Instance name* nel menu *Scheme*, o premere il tasto *FBD properties* nella barra *FBD* per cambiare il nome della ricorrenza del blocco funzione.

6.4.7 ACQUISIRE INFORMAZIONI SU UN BLOCCO

E' sempre possibile informarsi su un blocco aggiunto ad un documento FBD selezionandolo e compiendo una delle seguenti operazioni:

- Cliccare *Object>Open source* nel menu *Scheme*, o premere il tasto *View source* nella barra FBD, per aprire il codice sorgente di un blocco.
- Cliccare *Object properties* nel menu *Scheme*, o premere il tasto *FBD properties* nella barra *FBD*, per vedere le proprietà e i pins di input e output del blocco selezionato.

6.4.8 RILEVAZIONE AUTOMATICA DELL'ERRORE

L'editor FBD mostra automaticamente anche la posizione degli errori del compilatore. Per raggiungere il blocco dove l'errore ha avuto luogo, cliccare due volte sulla linea corrispondente dell'errore nella barra *Output*.

6.5 EDITOR SEQUENTIAL FUNCTION CHART (SFC)

L'editor SFC permette di i codificare e modificare le POU usando SFC (Sequential Function Chart) uno dei linguaggi ammessi in IEC.

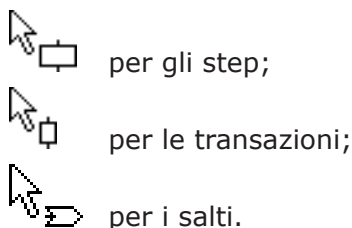
6.5.1 CREARE UN NUOVO DOCUMENTO SFC

Vedere la sezione "Creare e modificare le POU" (paragrafi 5.1.1 e 5.1.2).

6.5.2 INSERIRE UN NUOVO ELEMENTO SFC

Ci si può avvalere indifferentemente di una delle seguenti procedure:

- Cliccare Object>New nel menu Scheme, poi selezionare il tipo di nuovo elemento (azione, transizione o salto).
- Premere il tasto *New step*, *Add transition* o *Add jump* della barra SFC.

In ogni caso il puntatore del mouse cambia in:

6.5.3 CONNETTERE ELEMENTI SFC

Seguire la procedura seguente per connettere blocchi SFC:

- Cliccare *Connection mode* nel menu *Edit*, o premere il tasto *Connection* nella barra *FBD*, o premere semplicemente la barra spaziatrice della tastiera. Cliccare una volta sul pin sorgente, poi spostare il cursore sul pin di destinazione: l'editor SFC crea una connessione tra i due.
- Altrimenti, si può abilitare il modo di auto connessione cliccando *Autoconnect* nel menu *Scheme*, o cliccando il tasto *Autoconnect* nella barra *Network*. Prendere poi i due blocchi, e trascinarli uno vicino all'altro cosicché i pins rispettivi coincidano, ciò farà sì che l'editor SFC tracci automaticamente la connessione.

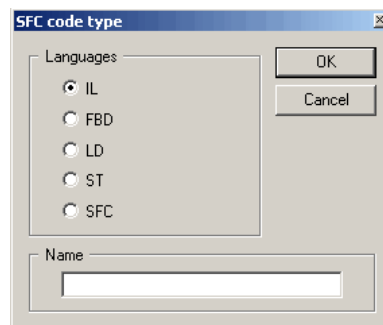
6.5.4 ASSEGNARE UN'AZIONE AD UNO STEP

Questo paragrafo spiega come rendere effettiva un'azione e come assegnarla ad uno stato.

6.5.4.1 SCRIVERE IL CODICE DI UN'AZIONE

Per iniziare l'implementazione di un'azione, occorre aprire un editor. Fare ciò seguendo una di queste istruzioni:

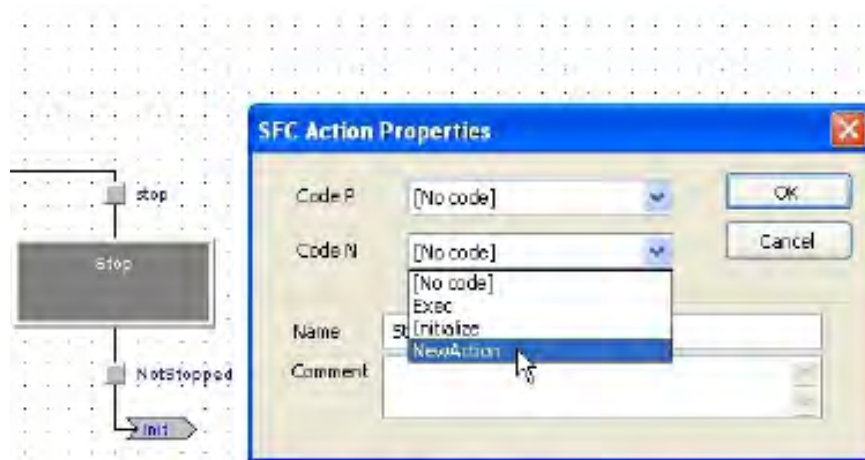
- Cliccare *Code object>New action* nel menu *Scheme*.
- Cliccare col tasto destro sul nome della POU SFC nella finestra *Workspace*. Apparirà un menu contestuale da cui si potrà scegliere il comando *New action*.

In tutti e due i casi, LogicLab mostra una finestra di dialogo come questa.

Selezionare uno dei linguaggi e digitare il nome della nuova azione nella casella di testo che si trova alla fine della finestra di dialogo. Confermare premendo *OK* o uscire premendo *Cancel*.

Premendo *OK*, LogicLab apre automaticamente l'editor associato al linguaggio selezionato nella finestra di dialogo precedente e si potrà digitare il codice della nuova azione.

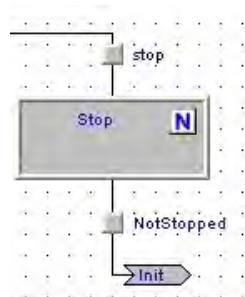
Notare che non è possibile dichiarare nuove variabili locali, poiché il modulo che si sta modificando è un componente del modulo SFC originale. La visibilità delle variabili locali si estende a tutte le azioni e transizioni che fanno parte del diagramma SFC.

6.5.4.2 ASSEGNARE UN'AZIONE AD UNO STEP

Una volta terminato di scrivere il codice, cliccare due volte sullo stato a cui si vuole assegnare il nuovo step. Apparirà una finestra di dialogo.

Dalla lista mostrata nel box *Code N,* selezionare, se lo step è attivo, il nome dell'azione che si vuole eseguire.

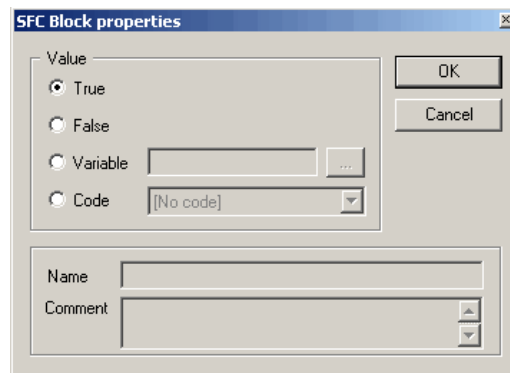
Si può anche scegliere, dalla lista mostrata nel box *Code P* (*Pulse*), il nome dell'azione che si vuole eseguire ogni volta che lo step diventa attivo (l'azione viene eseguita solo all'attivazione dello step, indipendentemente dal numero dei cicli degli step che rimangono attivi). Confermare l'assegnamento premendo *OK.*

Nello schema SFC, le assegnazioni di azioni a step sono rappresentate da lettere sull blocco dello step:

- azione *N* con la lettera N nell'angolo in alto a destra;
- azione P con la lettera P nell'angolo in basso a destra.

Se si avrà bisogno successivamente di modificare il codice sorgente dell'azione, basterà cliccare due volte su questo simbolo. Altrimenti, cliccare due volte sul nome dell'azione nella cartella *Actions* della finestra *Workspace*.

6.5.5 SPECIFICARE UNA COSTANTE/VARIABILE COME CONDIZIONE DI UNA TRANSIZIONE

Come affermato nella sezione del riferimento del linguaggio, una transizione può essere assegnata attraverso una costante, una variabile, un segmento di codice. Questo paragrafo spiega come usare i primi due, mentre il codice condizionale sarà trattato nel prossimo paragrafo.

Prima di tutto cliccare due volte sulla transizione a cui si vuole assegnare una condizione. Ciò farà apparire la seguente finestra di dialogo.

Selezionare *True* se si vuole che questa transizione sia costantemente verificata, *False* se si vuole che il programma PLC continui ad eseguire il blocco precedente.

Altrimenti, selezionando *Variable* la transizione dipenderà dal valore della variabile Booleana. Cliccare sulla parte corrispondente per rendere disponibile la casella di testo alla sua destra, e per specificare il nome della variabile.

Per questo scopo, è possibile servirsi dell'object browser, che si può richiamare premendo il tasto *Browse* mostrato qui sotto.

Premere *OK* per confermare o *Cancel* per abbandonare senza applicare i cambiamenti.

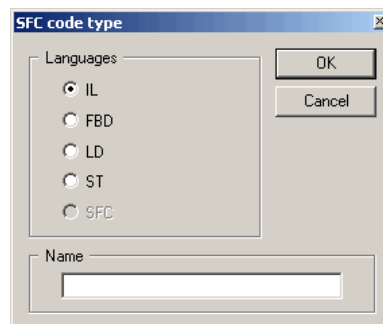
6.5.6 ASSEGNARE CODICE CONDIZIONALE AD UNA TRANSIZIONE

Questo paragrafo mostra come specificare una condizione attraverso un segmento di codice, e come assegnarlo ad una transizione.

6.5.6.1 SCRIVERE IL CODICE DI UNA TRANSIZIONE

Iniziare aprendo un editor, seguendo una delle seguenti procedure:

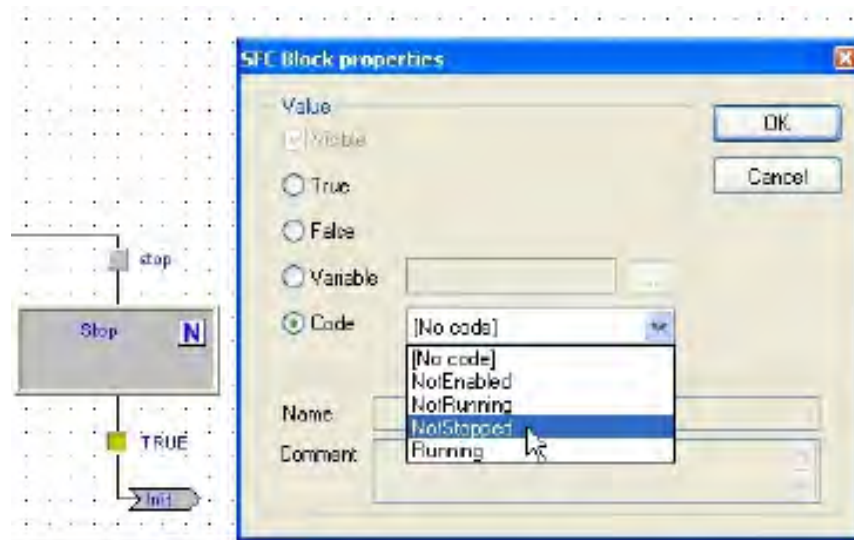
- cliccare *Code object>New transition* nel menu *Scheme*.
- Cliccare col tasto destro sul nome della POU SFC nella finestra *Workspace*, poi selezionare il comando *New transition* dal menu contestuale che apparirà.

In entrambi i casi, LogicLab mostra una finestra di dialogo come quella mostrata in figura.

Notare che si può usare qualunque linguaggio per codificare una condizione tranne SFC. Selezionare uno dei linguaggi e digitare il nome della nuova condizione nella casella di testo alla fine della finestra di dialogo. Poi confermare cliccando *OK* o abbandonare con *Cancel*.

Premendo *OK*, LogicLab apre automaticamente l'editor associato al linguaggio selezionato nella finestra di dialogo precedente e sarà possibile digitare il codice della nuova condizione.

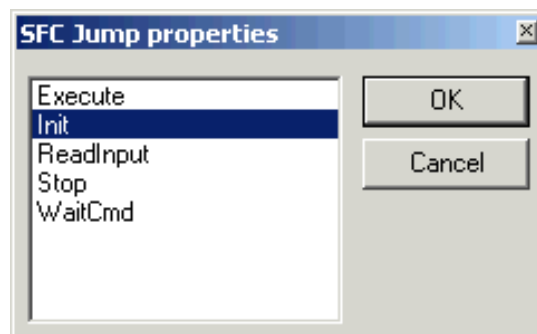
Notare anche che non è possibile dichiarare nuove variabili, poiché il modulo che si sta modificando fa parte del modulo SFC originale, che è la POU in cui possono essere dichiarate le variabili locali. La visibilità delle variabili locali si estenderà a tutte le azioni e transizioni che fanno parte del diagramma SFC.

6.5.6.2 ASSEGNARE UNA CONDIZIONE AD UNA TRANSIZIONE

Una volta terminato di scrivere il codice, cliccare due volte sullo stato a cui si vuole assegnare la nuova condizione. Apparirà una finestra di dialogo.

Selezionare il nome della condizione che si vuole assegnare a questo stato. Confermare premendo *OK*.

Se successivamente si avrà bisogno di modificare il codice sorgente della condizione, cliccare due volte sul nome della transizione nella cartella *Transitions* della finestra *Workspace*.

6.5.7 SPECIFICARE LA DESTINAZIONE DI UN JUMP

Per specificare lo step di destinazione di un jump, cliccare due volte sul blocco di jump nello spazio *Chart*. Ciò farà apparire una finestra di dialogo simile a quella mostrata sotto, che elenca il nome di tutti gli stati esistenti. Selezionare lo stato di destinazione, poi confermare premendo *OK* o abbandonare premendo *Cancel*.

6.5.8 MODIFICARE UN NETWORK SFC

L'editor SFC è fornito di funzioni comuni alla maggior parte di applicazione grafiche funzionanti sulla piattaforma Windows, ovvero:

- selezione di un blocco.
- Selezione di un set di blocchi premendo *CTRL+ tasto destro*.
- Operazioni di *Cut*,*Copy* e *Paste* sia di un singolo blocco sia di un set di blocchi.
- Drag-and-drop.

Alcune di queste funzioni sono accessibili tramite il menu *Edit* o la barra *Main*.

6.6 EDITOR DELLE VARIABILI

LogicLab comprende un editor grafico sia per le variabili globali che per quelle locali che fornisce un' interfaccia semplice per dichiarare e modificare le variabili: lo strumento si occupa della traduzione dei componenti di questi editor in codice sorgente IEC 61131-3 sintatticamente corretto.

Per esempio, considerare i componenti dell'editor variabili globali rappresentato nella seguente figura.

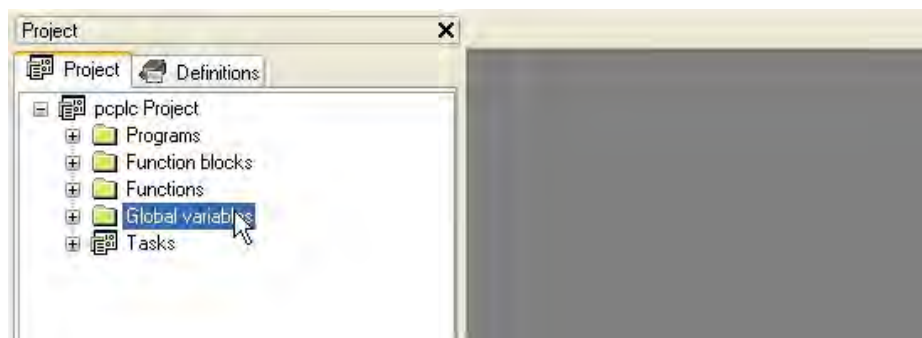
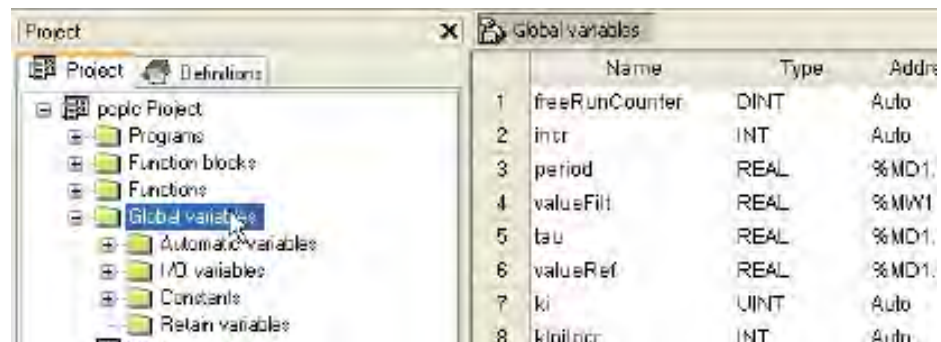
	Name	Type	Address	Group	Array	Init value	Attribute
9	gA	BOOL	Auto		No	TRUE	..
10	gB	REAL	Auto		[0..4]	5(0)	..
11	gC	REAL	%MD60.20		No	1.0	..
12	gD	INT	Auto		No	-74	CONSTANT

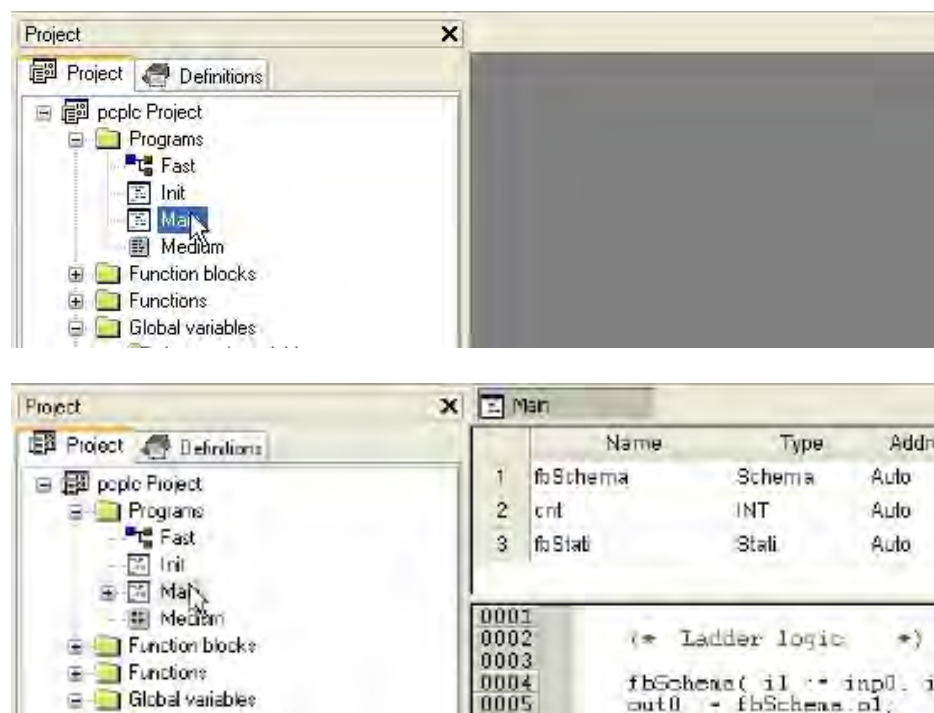
Il codice sorgente corrispondente avrà questo aspetto:

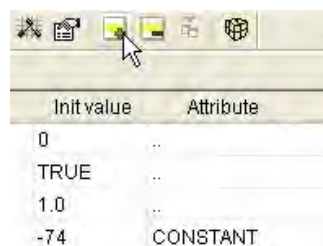
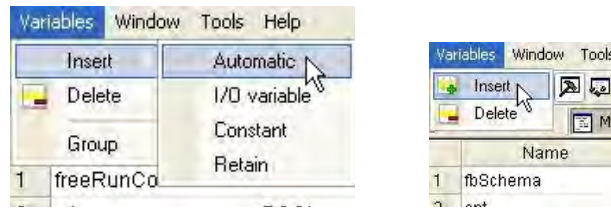
```
VAR_GLOBAL
      gA : BOOL := TRUE;
      gB : ARRAY[ 0..4 ] OF REAL;
      gC AT %MD60.20 : REAL := 1.0;
      END_VAR
      VAR_GLOBAL CONSTANT
      gD : INT := -74;
      END_VAR
```

6.6.1 APRIRE UN EDITOR DI VARIABILI

6.6.1.1 APRIRE L'EDITOR DELLE VARIABILI GLOBALI

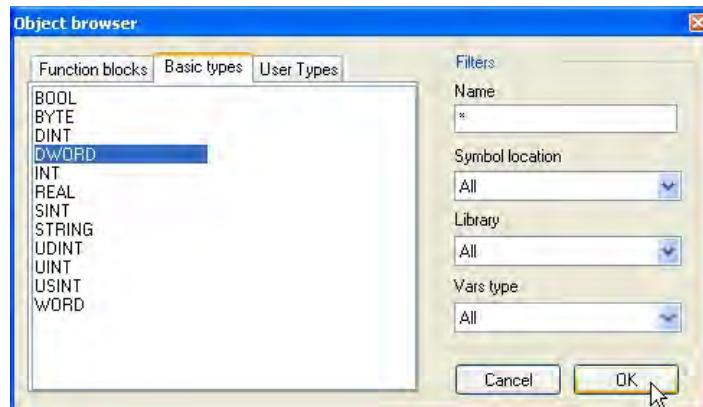
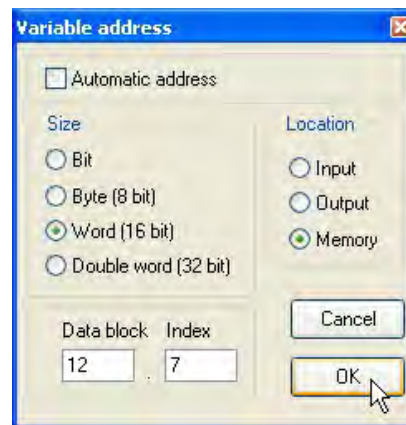
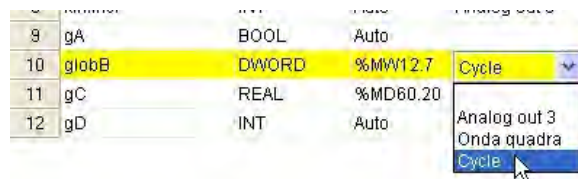
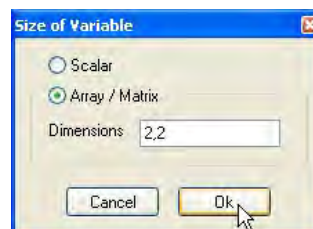
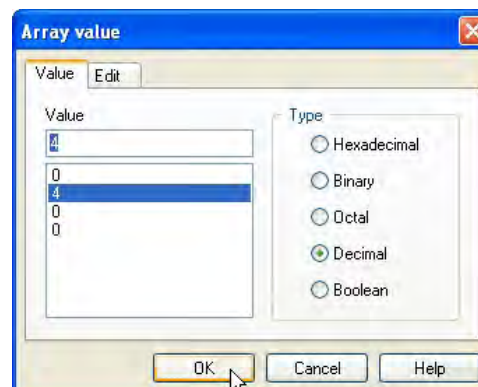
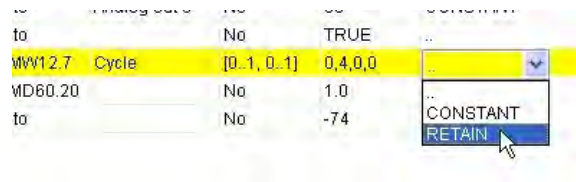
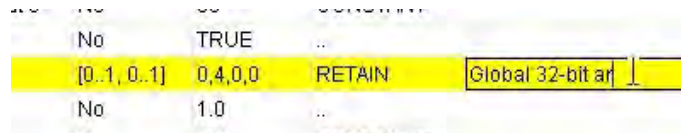
Per aprire l'editor variabili globali, cliccare due volte su *Global variables* nell'albero del progetto.

6.6.1.2 APRIRE L'EDITOR DELLE VARIABILI LOCALI

Per aprire l'editor variabili locali, aprire semplicemente la POU di cui le variabili che si desidera modificare sono locali.

6.6.2 CREARE UNA NUOVA VARIABILE

Per creare una nuova variabile, si può cliccare sulla voce *Insert record* nella barra *Project*.

Altrimenti, accedere al menu *Variables* e scegliere *Insert*.

6.6.3 MODIFICARE LE VARIABILI

Seguire la procedura seguente per modificare la dichiarazione di una variabile in un editor di variabili (tutti i passaggi seguenti sono opzionali e normalmente si salteranno durante la modifica di una variabile):

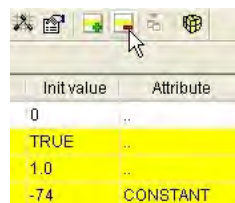
1) Modificare il nome della variabile inserendo il nuovo nome nella cella corrispondente.

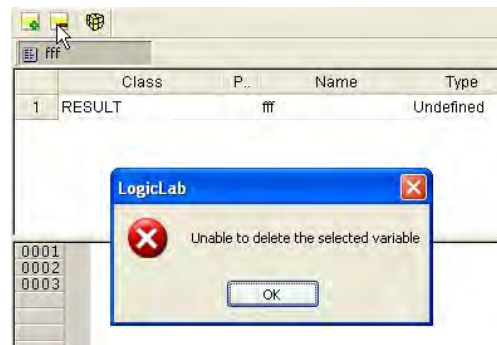
2) Cambiare il tipo di variabile, editando il nome del tipo nella cella corrispondente o cliccare sul pulsante e selezionare il tipo desiderato dall'elenco che compare.

3) Modificare l'indirizzo della variabile cliccando sul tasto nella cella corrispondente e inserendo le informazioni richieste nella finestra che compare. Notare che, in caso di variabili globali, questa operazione potrebbe cambiare la posizione della variabile nell'albero del progetto.

4) Nel caso di variabili globali, è possibile assegnare la variabile ad un gruppo, selezionandolo dalla lista che si apre cliccando sulla cella corrispondente. Questa operazione cambierà la posizione della variabile nell'albero del progetto.

5) Scegliere se la variabile è un array o no; se lo è, modificare la dimensione della variabile.

6) Modificare i valori iniziali della variabile: cliccare sul tasto nella cella corrispondente e inserire i valori nella finestra che compare.

7) Assegnare un attributo alla variabile (per esempio, `CONSTANT` o `RETAIN`), selezionandolo dalla lista che si apre cliccando sulla cella corrispondente.

8) Digitare la descrizione della variabile nella cella corrispondente. Notare che, nel caso di variabili globali, questa operazione potrebbe cambiare la posizione della variabile nell'albero del progetto.

9) Salvare il progetto per mantenere i cambiamenti fatti alla dichiarazione della variabile.

6.6.4 CANCELLARE LE VARIABILI

Per cancellare una o più variabili, selezionarle nell'editor: si possono usare i tasti *CTRL* o *SHIFT* per selezionare più elementi.

	Name	Type	Address
1	freeRunCounter	DINT	Auto
2	gA	BOOL	Auto
3	gC	REAL	%MD60.20
4	gD	INT	Auto
5	ki	UINT	Auto
6	incr	INT	Auto
7	kIniIncr	INT	Auto
8	globB	DWORD	%MW12.7
9	period	REAL	%MD1.11
10	tau	REAL	%MD1.9
11	valueFilt	REAL	%MW1.8

Cliccare poi su *Delete* nella barra *Project*.

Altrimenti, si può accedere al menu *Variables* e scegliere *Delete*.

Notare che non si può cancellare il `RESULT` di una `FUNCTION IEC61131-3`.

6.6.5 ORDINARE LE VARIABILI

E’ possibile ordinare le variabili nell’editor cliccando sull’intestazione della colonna del campo che si desidera utilizzare come criterio di ordinamento.

	Name	Type	Ac
1	valueFilt	REAL	%MV
2	tau	REAL	%MI
3	valueRef	REAL	%MI
4	period	REAL	%MI
5	kInilncr	INT	Auto
6	ki	UINT	Auto
7	incr	INT	Auto
8	freeRunCounter	DINT	Auto

	Name	Type	Ac
1	freeRunCounter	DINT	Auto
2	incr	INT	Auto
3	ki	UINT	Auto
4	kInilncr	INT	Auto
5	period	REAL	%MI
6	tau	REAL	%MI
7	valueFilt	REAL	%MV
8	valueRef	REAL	%MI

6.6.6 COPIARE LE VARIABILI

L’editor variabili permette di copiare e incollare elementi rapidamente. Si possono usare le scorciatoie della tastiera o accedere a queste componenti dal menu *Edit*.

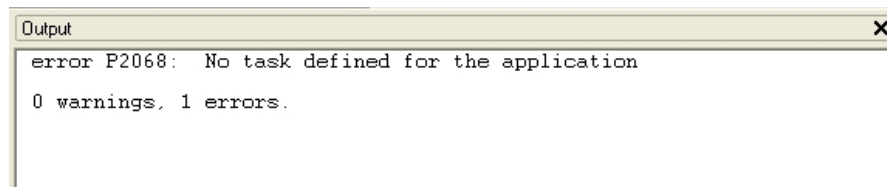
	Name	Type	Address	
1	freeRunCounter	DINT	Auto	
2	incr	INT	Auto	An
3	ki	UINT	Auto	
4	kInilncr	INT	Auto	An
5	period	REAL	%MD1.11	Or
6	tau	REAL	%MD1.9	Or
7	valueFilt	REAL	%MW1.8	Or
8	valueRef	REAL	%MD1.10	Or
9	tau1	REAL	%MD1.9	Or

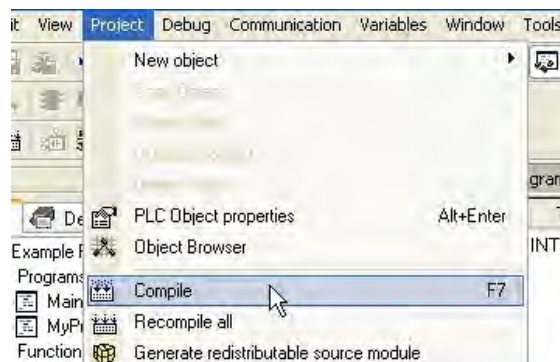
7. COMPILARE

La compilazione consiste nel prendere il codice sorgente PLC e tradurlo automaticamente in codice binario, che può essere eseguito dal processore del target device.

7.1 COMPILARE IL PROGETTO

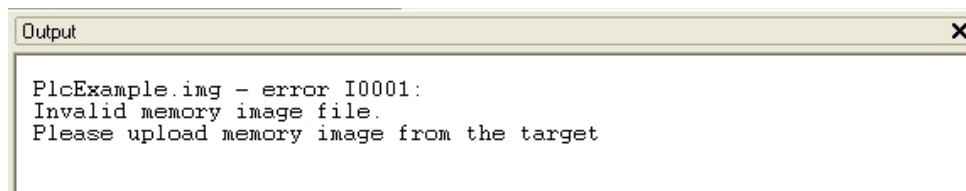
Prima di iniziare la compilazione effettiva, assicurarsi che almeno un programma sia stato assegnato ad un task.

Quando non sussiste questa pre-condizione, la compilazione genera un messaggio di errore significativo.

Per cominciare la compilazione, cliccare su *Compile* nella barra *Project*.

Altrimenti, si può scegliere *Compile* dal menu *Project* o premere *F7* sulla testiera.

LogicLab salva automaticamente tutti i cambiamenti al progetto prima di iniziare la compilazione.

7.1.1 CARICARE UN FILE IMMAGINE

Prima di compiere la compilazione effettiva, il compilatore necessita di caricare il file immagine (img file) che contiene la mappa di memoria del target device. Se il target è connesso al momento dell'inizio della compilazione, il compilatore cercherà il file immagine direttamente nel target stesso. Altrimenti, caricherà la copia locale del file immagine dalla cartella di lavoro. Se il target device non è connesso e non c'è copia locale del file immagine, la compilazione non può avvenire: sarà necessario connettersi ad un target device operativo.

```
Output
PlcExample.img - error I0001:
Invalid memory image file.
Please upload memory image from the target
```

7.2 OUTPUT DEL COMPILATORE

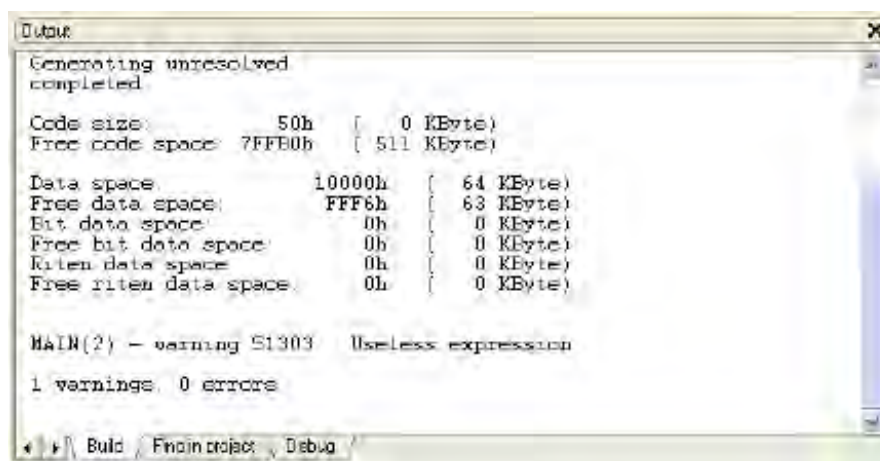
Se è stato completato il passaggio precedente, il compilatore compie la compilazione effettiva e stampa un report nella barra *Output*. L'ultima stringa del report ha il formato seguente:

```
m warnings, n errors
```

Informa l'utente del risultato della compilazione.

Condizione	Descrizione
n>0	Errori di compilazione. Il codice PLC contiene uno o più errori gravi, che non possono essere aggirati dal compilatore.
n=0, m>0	Warning di compilazione. Il codice PLC contiene uno o più errori minori, che il compilatore individua e aggira automaticamente. Si è comunque informati che il programma PLC può comportarsi in una maniera diversa da quella che ci si aspetta: si è spronati ad eliminare questi warning modificando e ricompilando l'applicazione finché non appare nessun messaggio di warning.
n=m=0	Il codice PLC è interamente corretto, la compilazione è eseguita. Si dovrebbe sempre lavorare con 0 warning, 0 error.

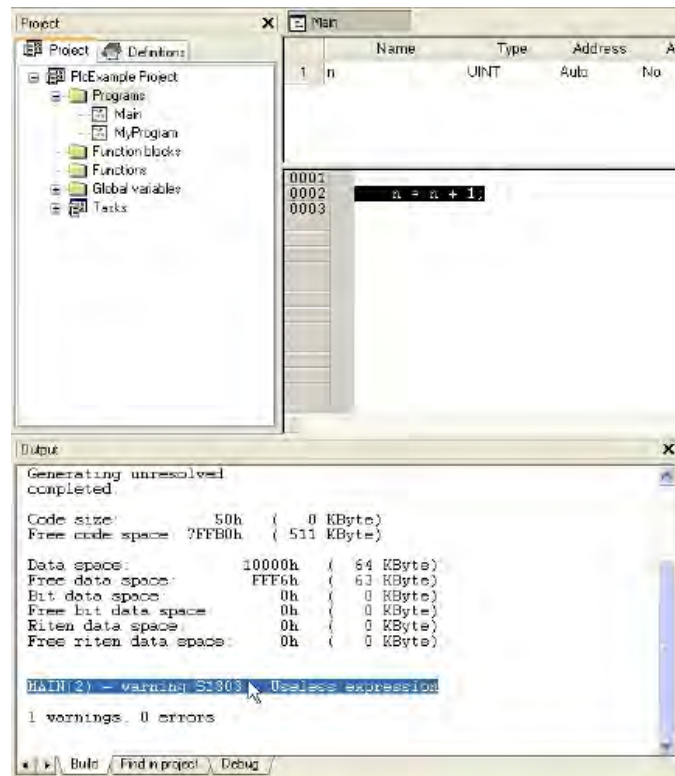
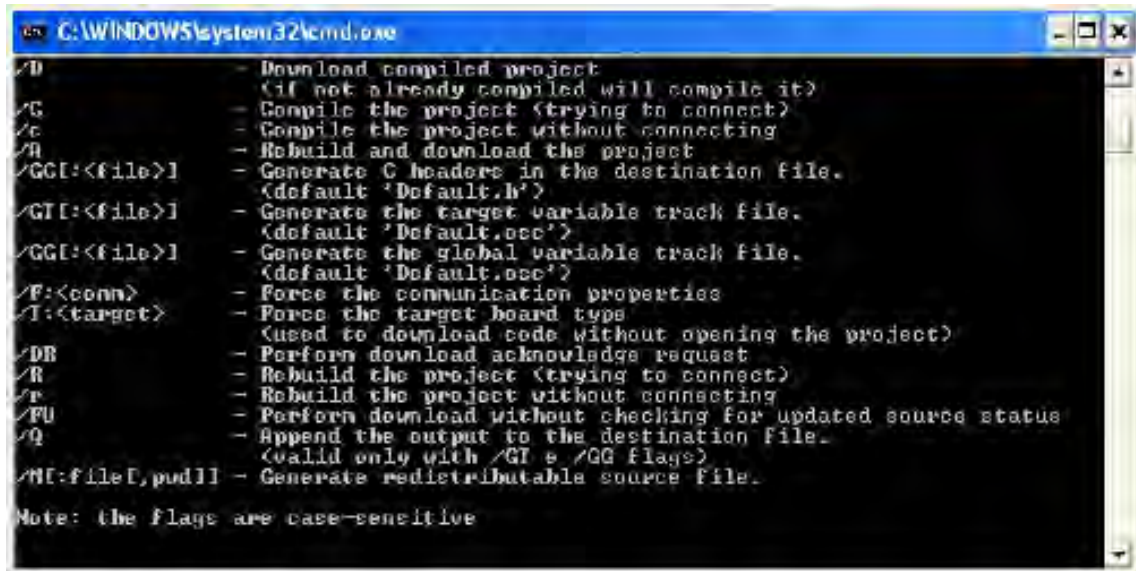
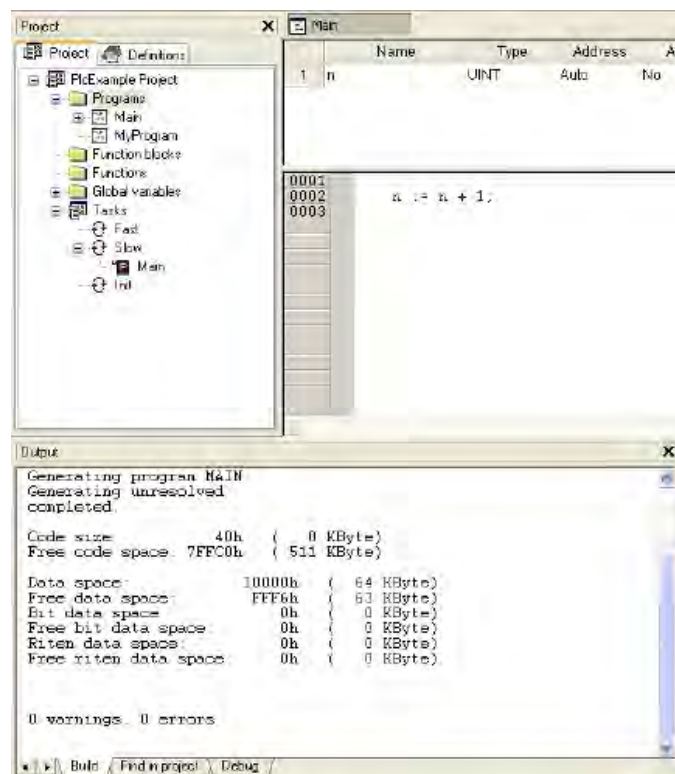
7.2.1 ERRORI DEL COMPILATORE

Quando l'applicazione contiene uno o più errori, nella finestra *Output* vengono riportate alcune preziose informazioni per ognuno di questi errori.

Come si può vedere, l'informazione comprende:

- il nome della POU che presenta l'errore;
- il numero della linea del codice sorgente in cui l'errore si è verificato;
- se si tratta di un errore bloccante (`error`) o di un warning che il compilatore può aggirare (`warning`);
- il codice dell'errore;

- la descrizione dell'errore.

Fare riferimento all'apposita sezione per la descrizione degli errori del compilatore.

Cliccando due volte sul messaggio di errore nella barra *Output*, LogicLab apre il codice sorgente ed evidenzia la linea contenente l'errore.

Sarà quindi possibile risolvere il problema e ricompilare.

7.3 COMPILATORE COMMAND-LINE

Il compilatore di LogicLab può essere usato indipendentemente dall'IDE: nella directory LogicLab, si trova un file eseguibile chiamato "Command-line compiler", che può essere richiamato (per esempio in un file batch) con numerose opzioni.

Per ottenere le informazioni sulla sintassi e le opzioni di questo strumento di command-line, lanciare il file eseguibile senza parametri.

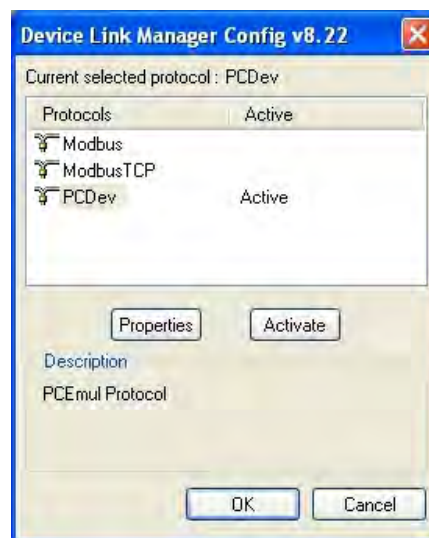
8. LANCIARE L'APPLICAZIONE

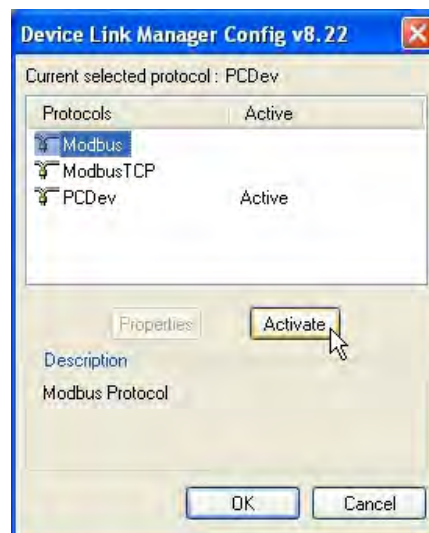
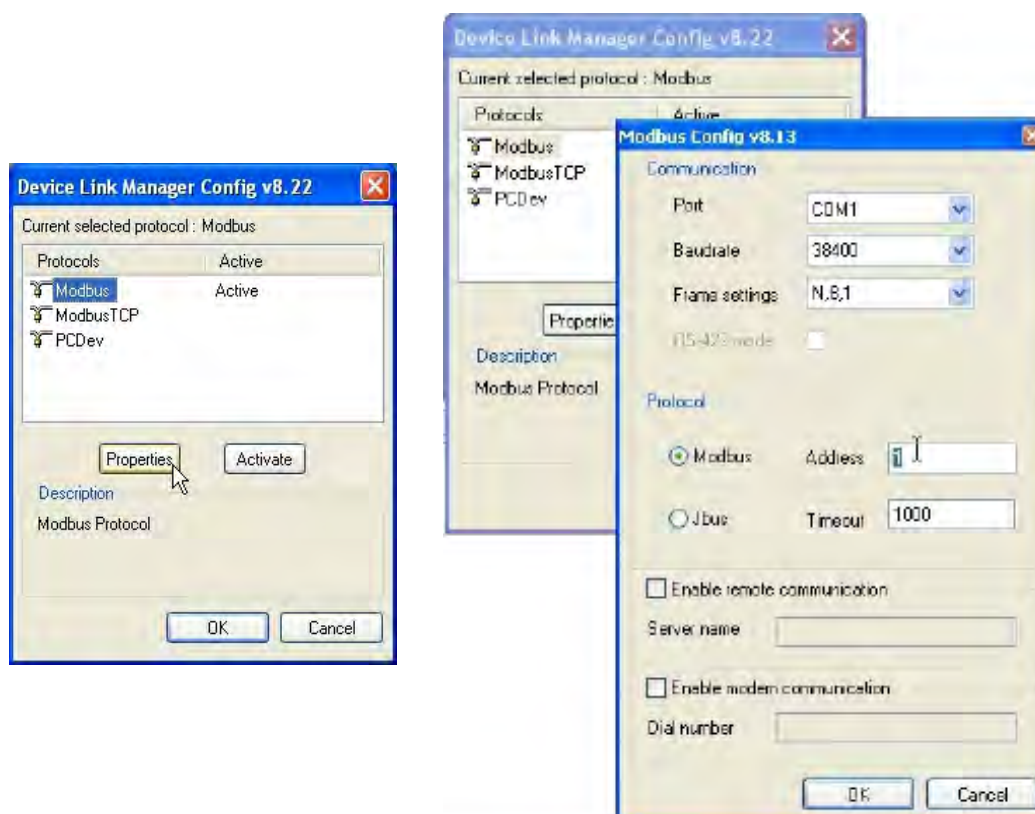
Per eseguire il download e il debug dell'applicazione, è necessario stabilire una connessione col target device. Questo capitolo si focalizza sulle operazioni richieste per connettersi al target device ed eseguire il download dell'applicazione, mentre l'ampia gamma di strumenti di debug verranno descritti in un capitolo separato (vedi capitolo 9).

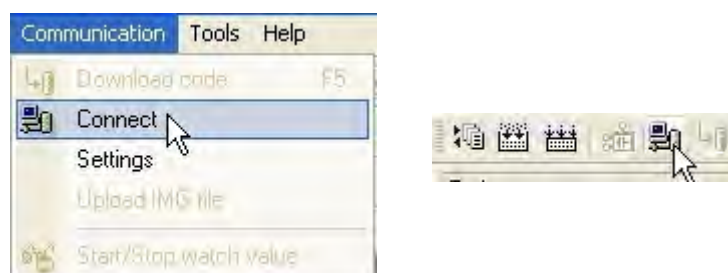
8.1 IMPOSTARE LA COMUNICAZIONE

Per stabilire una comunicazione col target device, assicurarsi che sia collegato correttamente (tutti i cavi devono essere attaccati, il network configurato correttamente, ecc…).

Seguire la procedura seguente per impostare e stabilire la connessione al target device:

1) Cliccare su `Settings` nel menu `Communication` della finestra principale di LogicLab. Questo farà apparire la seguente finestra di dialogo.

Gli elementi presenti nella lista dei protocolli di comunicazione dipendono dai file eseguibili di setup fatti funzionare sul PC (riferirsi al fornitore dell'hardware se un protocollo che doveva essere presente nella lista non compare).

2) Selezionare il protocollo appropriato e renderlo attivo premendo il bottone `Activate`.

3) Compilare tutti le impostazioni specifiche del protocollo (per esempio l'indirizzo del timeout di comunicazione – ovvero per quanto tempo LogicLab deve aspettare una risposta dal target prima di riportare un messaggio di errore).

4) Applicare i cambiamenti apportati alle impostazioni di comunicazione.

E' ora possibile stabilire la comunicazione cliccando su `Connect` nel menu `Communication`, o premendo il tasto `Connect` nella barra `Project`.

8.1.1 SALVARE L'ULTIMA PORTA DI COMUNICAZIONE USATA

Quando si connette un target device usando una porta seriale (porta COM), di solito si usa sempre la stessa porta per tutti i device (molti PC moderni hanno una sola porta COM).

Potete salvare l'ultima porta COM utilizzata è lasciare che LogicLab usi questa porta per sovrascrivere le impostazioni del progetto: questa funzione si rileva particolarmente utile quando si condividono progetti con altri sviluppatori, che possono usare porte COM differenti per connettersi al target device.

Per salvare le impostazioni della porta COM, abilitare l'opzione *Use last port* nel menu *File > Options*.

8.2 STATO ON-LINE

8.2.1 STATO DELLA CONNESSIONE

Lo stato della comunicazione è mostrato in una piccola casella vicina al bordo destro della barra *Status*.

Se non si ha ancora provato a stabilire una connessione, lo stato della comunicazione è impostato su *Not connected*.

Provando a connettersi al target device, lo stato di comunicazione diventa uno dei seguenti:

- `Error`: la comunicazione non può essere stabilita, controllare sia i link fisici che le impostazioni di comunicazione.

- `Connected`: la comunicazione è stata stabilita.

8.2.2 STATO DELL'APPLICAZIONE

Vicino allo stato della comunicazione c'è un altro piccolo riquadro che fornisce informazioni sullo stato dell'applicazione attualmente in esecuzione sul target device.

Quando lo stato di connessione è `Connected`, lo stato dell'applicazione assume uno dei seguenti valori.

- `No code`: nessuna applicazione è eseguita sul target device.

- `Diff. code`: l’applicazione attualmente in esecuzione sul target device non è la stessa di quella aperta nell’IDE; inoltre non è disponibile nessuna informazione di debug in linea con l’applicazione: i valori mostrati nella finestra watch o nell’oscilloscopio non sono reali e la modalità debug non può essere attivata.

DIFF. CODE

- `Diff. code, Symbols OK`: l’applicazione attualmente in esecuzione sul target device non è la stessa di quella attualmente aperta nell’IDE; tuttavia sono disponibili alcune informazioni di debug (per esempio, perché l’applicazione è stata precedentemente scaricata dal target device dallo stesso PC): il valore mostrato nella finestra watch o nell’oscilloscopio sono reali, ma la modalità debug non può essere ancora attivata.

DIFF. CODE (SYM)

- `Source OK`: l’applicazione attualmente in esecuzione sul traget device è la stessa attualmente aperta sull’IDE: la modalità debug può essere attivata.

8.3 ESEGUIRE IL DOWNLOAD DELL’APPLICAZIONE

Un’ applicazione PLC compilata deve essere scaricata al target device in modo che il processore la esegua. Questo paragrafo mostra come inviare un codice PLC ad un target device. Notare che LogicLab può eseguire il download del codice al target device solo se quest’ultimo è collegato al PC su cui LogicLab sta funzionando. Vedere la sezione corrispondente per i dettagli.

Per eseguire il download dell’applicazione, cliccare sul tasto corrispondente nella barra *Project*.

In alternativa, scegliere *Download code* dal menu *Project* o premere il tasto *F5*.

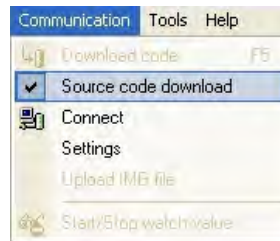
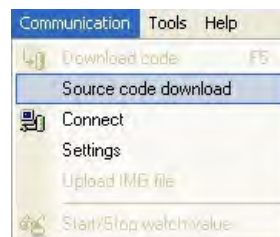
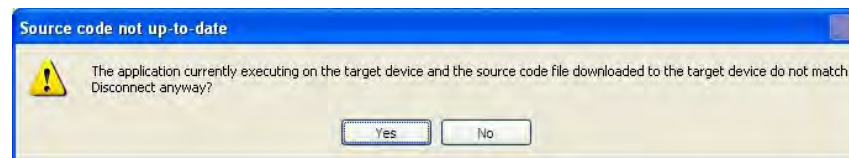
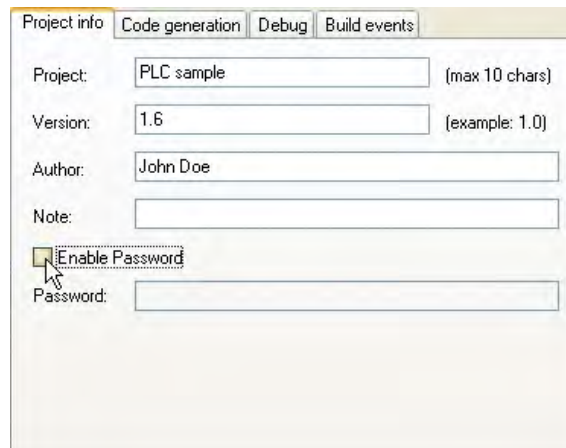
LogicLab controlla che il progetto non abbia dei cambiamenti non salvati. In questo caso, comincia la compilazione dell’applicazione automaticamente. Alla fine il codice binario è inviato al target device, che esegue un reset automatico alla fine della trasmissione. Ora il codice inviato è effettivamente eseguito dal processore del target device.

8.3.1 CONTROLLARE IL CODICE SORGENTE SCARICATO

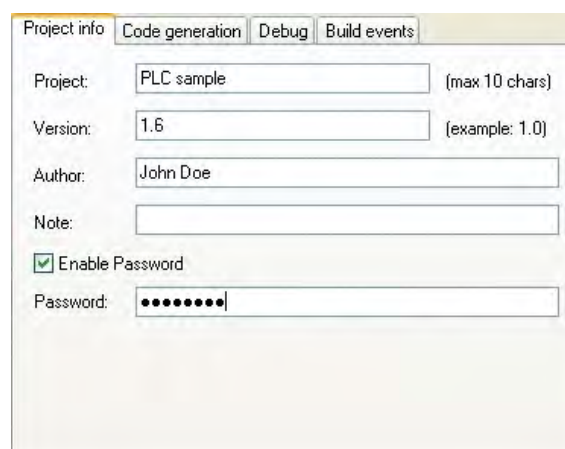
Se il codice sorgente dell’applicazione viene scaricato insieme al codice binario o no, dipende dal target device con cui ci si sta interfacciando: alcuni device ospitano il codice sorgente dell’applicazione nella loro memoria, in modo di permettere allo sviluppatore di caricare il progetto in un momento successivo.

In questo caso, potete controllare alcuni aspetti del processo di download del codice sorgente, come spiegato nel paragrafo seguente.

8.3.1.1 INTERROMPERE IL DOWNLOAD DEL CODICE SORGENTE

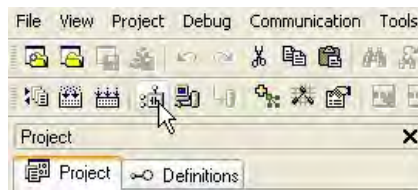
Al fine di accelerare il ciclo di sviluppo, si consiglia di disabilitare il download del codice sorgente: deselezionare la voce `Source code download` nel menu `Communication`.

Quando si ferma lo sviluppo dell'applicazione, è possibile riattivare il download del codice sorgente cliccando sulla stessa voce del menu.

Quando ci si disconnette dal target device, LogicLab controlla se l'applicazione attualmente in esecuzione sul target e il codice sorgente disponibile a bordo coincidono, avvisandovi se non lo sono.

8.3.1.2 PROTEGGERE IL CODICE SORGENTE CON UNA PASSWORD

Si consiglia di proteggere il codice sorgente scaricato sul target device con una password, così che LogicLab non aprirà il progetto scaricato se la pasword inserita non sarà corretta.

Aprire la finestra `Project options` (menu ... `Project > Options`) e impostare la password.

Si può anche scegliere di disabilitare la password.

8.4 SIMULAZIONE

A seconda del target con cui ci si sta interfacciando, si può essere in grado di simulare l'esecuzione dell'applicazione PLC con l'ambiente di simulazione integrato di LogicLab: SimuLab.

Per avviare la simulazione, basta cliccare sulla voce corretta della barra *Project*.

Fare riferimento al manuale di <<Simulation>> per ottenere informazioni su come controllare la simulazione.

9. DEBUG

LogicLab fornisce numerosi strumenti di debug, che aiutano lo sviluppatore a controllare se l'applicazione si comporta come voluto oppure no.

Tutti questi strumenti di debug fondamentalmente permettono allo sviluppatore di vedere il valore delle variabili selezionate mentre l'applicazione PLC sta funzionando.

Gli strumenti di debug di LogicLab possono essere raggruppati in due classi:

- Strumenti di debug asincroni. Leggono i valori delle variabili selezionate dallo sviluppatore con richieste successive inviate al target device. Tanto gli strumenti di debug (in esecuzione sul PC) quanto, potenzialmente, il task responsabile di rispondere a queste richieste (sul target device) eseguono indipendentemente dall'applicazione PLC. Quindi non ci sono garanzie di sincronismo, in relazione all'esecuzione dell'applicazione PLC, sui valori di due distinte variabili campionate allo stesso momento (uno o più cicli PLC potrebbero intercorrere tra il campionamento della prima variabile e quello della seconda); per la stessa ragione, l'evoluzione del valore di una singola variabile non è affidabile, soprattutto quando cambia velocemente.
- Strumenti di debug sincroni. Richiedono la definizione di un trigger nel codice PLC. Aggiornano simultaneamente tutte le variabili a loro assegnate ogni volta che l'esecuzione dell'applicazione PLC raggiunge il trigger, poiché nessun'altra istruzione può essere eseguita finché il valore di ogni variabile è stato aggiornato. Come risultato, gli strumenti di debug sincroni sopperiscono alle limitazioni caratteristiche di quelli asincroni.

Questo capitolo mostra come eseguire il debug dell'applicazione usando sia gli strumenti sincroni che quelli asincroni.

9.1 FINESTRA WATCH

La finestra *Watch* permette di monitorare i valori attuali di un set di variabili. Essendo uno strumento asincrono, la finestra *Watch* non garantisce la sincronizzazione dei valori. Per questo motivo, leggendo i valori delle variabili nella finestra *Watch* è bene ricordare che sussiste la possibilità che possano riferirsi a cicli di esecuzione differenti dal task corrispondente.

La finestra *Watch* contiene una voce per ciascuna variabile che abbiamo aggiunto ad essa. Le informazioni mostrate nella finestra *Watch* comprendono il nome della variabile, il suo valore, il suo tipo e la sua posizione nell'applicazione PLC.

Symbol	Value	Type	Location
incr	50	INT	@Slow:Main
fbSchema.i2	TRUE	BOOL	@Slow:Main
fbPulse.enable	FALSE	BOOL	@Fast:Fast
incr	50	INT	@Init:Init

9.1.1 APRIRE E CHIUDERE LA FINESTRA WATCH

Per aprire la finestra *Watch*, cliccare sul bottone *Watch* della barra *Main*.

Per chiudere la finestra *Watch*, cliccare ancora sul bottone *Watch*.

In alternativa, cliccare sul bottone *Close* sull'angolo in alto a destra della finestra *Watch*.

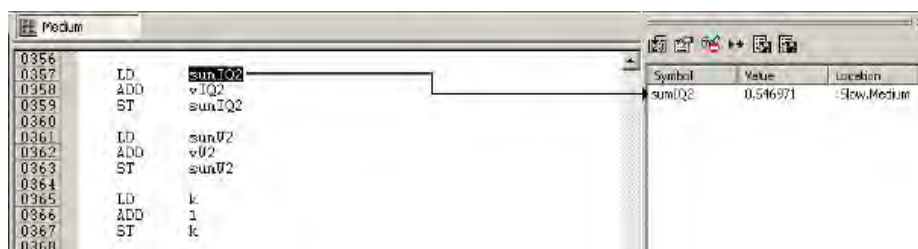
In entrambi i casi, chiudere la finestra *Watch* significa semplicemente nasconderla, non reimpostarla. A prova di ciò, chiudendo e riaprendo la finestra *Watch* si noterà che contiene sempre le variabili aggiunte ad essa.

9.1.2 AGGIUNGERE OGGETTI ALLA FINESTRA WATCH

Per vedere una variabile, occorre prima aggiungerla alla lista watch.

Notare che, a differenza delle finestre trigger e della finestra *Graphic trigger*, è possibile aggiungere alla finestra *Watch* tutte le variabili del progetto, senza preoccuparsi di dove sono dichiarate.

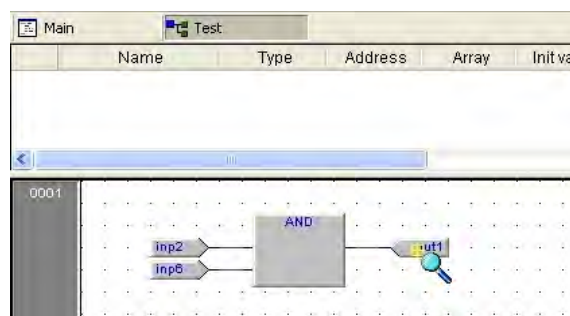
9.1.2.1 AGGIUNGERE UNA VARIABILE DA UN EDITOR DI CODICE SORGENTE PER I LINGUAGGI TESTUALE

Seguire la procedura seguente per aggiungere una variabile da un editor di codice sorgente testuale (come, IL or ST) alla finestra *Watch*: selezionare una variabile cliccando due volte su di essa e trascinarla nella finestra *Watch*.

La stessa procedura funziona per tutte le variabili che si vogliono inserire.

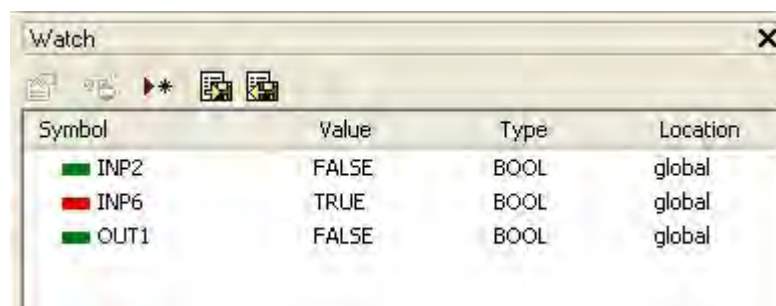
9.1.2.2 AGGIUNGERE UNA VARIABILE DA UN EDITOR DI CODICE SORGENTE PER I LINGUAGGI GRAFICI

Seguire la procedura seguente per aggiungere una variabile da un editor di codice sorgente grafico (LD,FBD o SFC) alla finestra *Watch*:

1) Premere il bottone *Watch* sulla barra FBD.

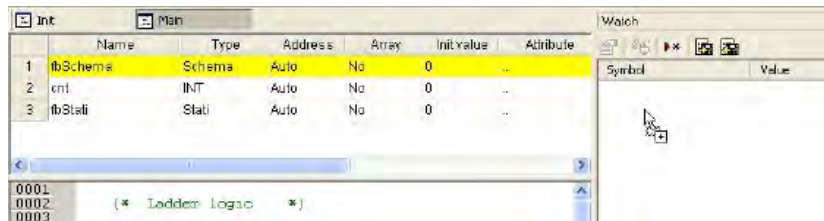
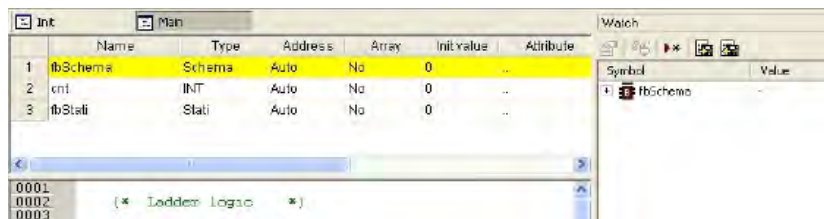
2) Cliccare sul blocco che rappresenta la variabile che si vuole che sia mostrata nella finestra *Watch*.

3) Apparirà una finestra di dialogo che elenca tutte le ricorrenze di finestre di debug attualmente esistenti, chiedendo quale debba ricevere l'oggetto sul quale si ha appena cliccato.

Per mostrare la variabile nella finestra *Watch*, premere *Watch* e poi *OK*.

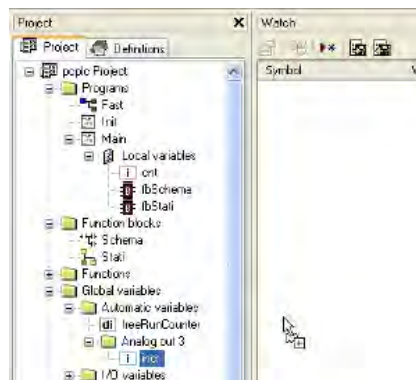
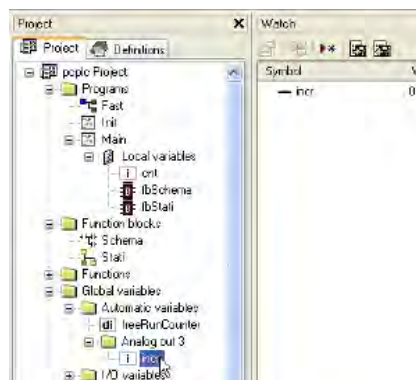
Il nome della variabile, il valore e la locazione sono ora visualizzate in una nuova riga della finestra *Watch*.

La stessa procedura si applica a tutte le variabili che si vogliono controllare.

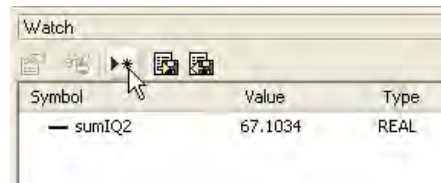
Una volta inserite nella finestra *Watch* tutte le variabili che si vogliono visualizzare, cliccare sul tasto *Select/Move* della barra FBD: il cursore tornerà alla sua forma originale.

9.1.2.3 AGGIUNGERE UNA VARIABILE DA UN EDITOR DI VARIABILI

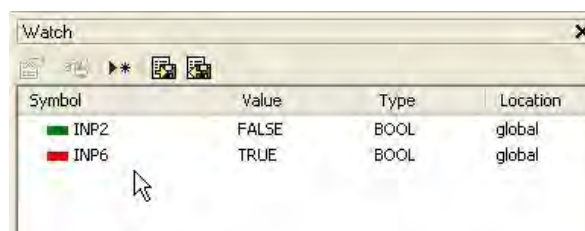
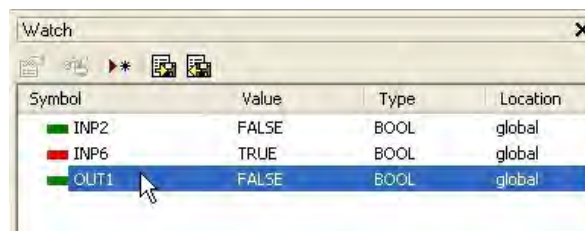
Per aggiungere una variabile alla finestra *Watch*, selezionare il record corrispondente dell'editor variabili e fare drag-and-drop nella finestra *Watch*

o premere *F8*.

9.1.2.4 AGGIUNGERE UNA VARIABILE DALL'ALBERO DEL PROGETTO

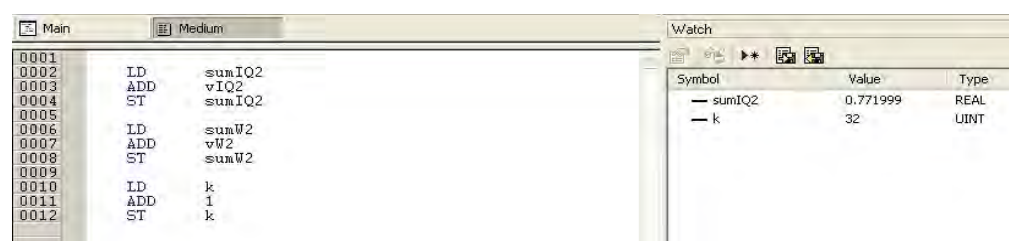
Per aggiungere una variabile alla finestra *Watch*, selezionarla nell'albero del progetto e fare drag-and-drop nella finestra *Watch*

o premere *F8*.

9.1.2.5 AGGIUNGERE UNA VARIABILE DELLA BARRA DELLA FINESTRA WATCH

Per aggiungere una variabile è anche possibile cliccare sull'apposita voce della barra interna della finestra *Watch*.

Digitare (o selezionare navigando i simboli del progetto) il nome della variabile e la sua posizione (dove è stata dichiarata).

9.1.3 RIMUOVERE UNA VARIABILE

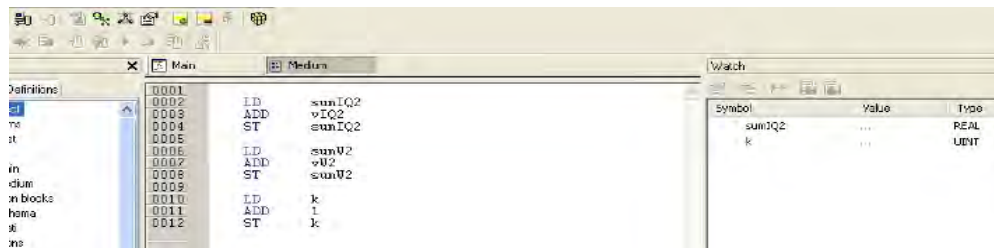
Affinché una variabile non sia più mostrata nella finestra *Watch*, selezionarla cliccando sul suo nome e premere il tasto *Del*.

9.1.4 AGGIORNAMENTO DEI VALORI

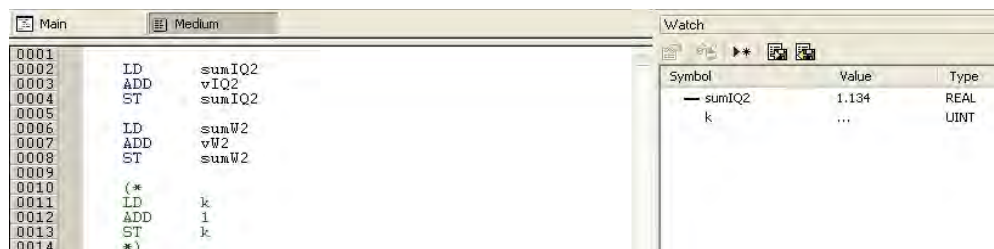
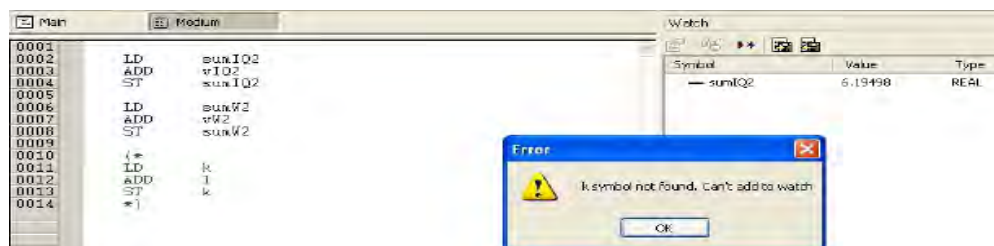
9.1.4.1 OPERAZIONI NORMALI

Considerare l'esempio seguente.

Il manager della finestra *Watch* legge periodicamente il valore delle variabili dalla memoria.

Tuttavia, quest'azione avviene in modo asincrono, quindi può succedere che un task a priorità più alta modifichi il valore di alcune delle variabili mentre queste sono lette. Quindi, alla fine del processo di aggiornamento, i valori mostrati nella finestra potrebbero riferirsi a stati di esecuzione diversi del codice PLC.

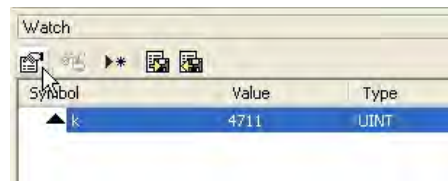
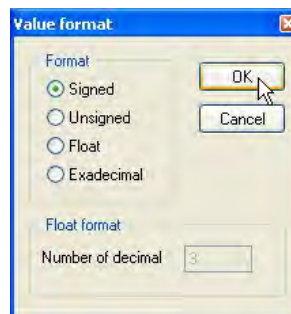
9.1.4.2 TARGET DISCONNESSO

Se il target device è sconnesso, la colonna *Value* contiene tre puntini.

9.1.4.3 OGGETTO NON TROVATO

Se il codice PLC cambia e LogicLab non riesce a riportare la posizione di memoria di un oggetto nella finestra *Watch*, la colonna *Value* conterrà tre puntini.

Provando ad aggiungere alla finestra *Watch* un simbolo che non è stato allocato, LogicLab darà il seguente messaggio d'errore.

9.1.5 CAMBIARE IL FORMATO DI UN DATO

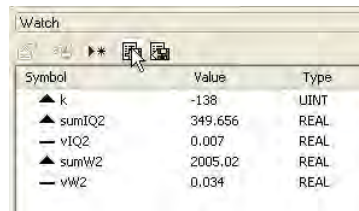
Quando si aggiunge una variabile alla finestra *Watch*, LogicLab riconosce automaticamente il suo tipo (intero senza segno, intero con segno, virgola mobile, esadecimale) e mostra regolarmente il suo valore. Inoltre, se la variabile è a virgola mobile , LogicLab gli assegna un numero decimale di default.

Tuttavia, ci potrebbe essere bisogno di stampare la variabile in un formato differente.

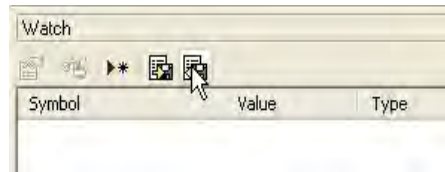
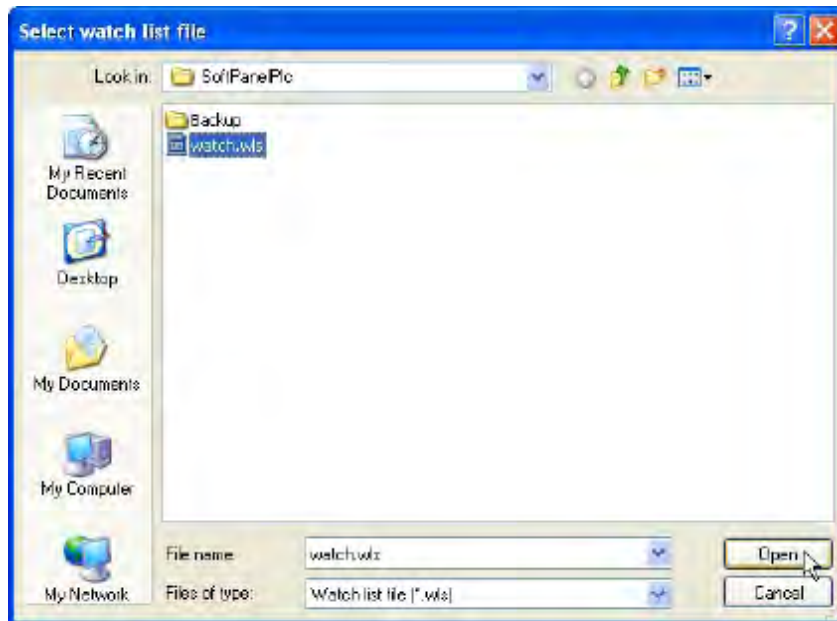
Per impostare un altro formato diverso da quello assegnato da LogicLab, premere il tasto *Format value* della barra di applicazione.

Scegliere il formato e confermare l'operazione.

9.1.6 LAVORARE CON UNA WATCH LIST

E' possibile salvare come file il set delle voci nella finestra *Watch*, per poter facilmente ri-memorizzare lo status di questo strumento di debug in una sessione di lavoro successiva.

Per salvare l'elenco, seguire questa procedura:

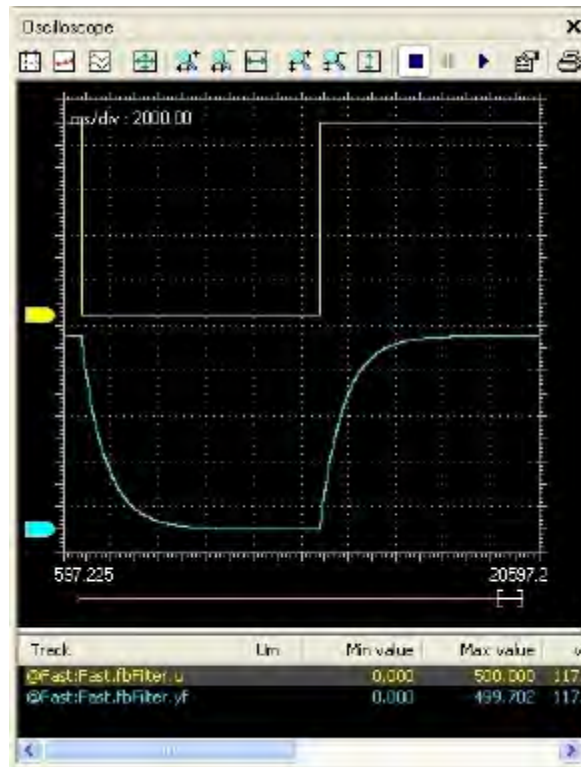
1) Cliccare sulla voce corrispondente nella barra *Watch window.*

2) Inserire il nome del file e scegliere la sua destinazione nel file system.

Per caricare una watch list, procedere come segue:

1) Cliccare sulla voce corrispondente della barra *Watch window*.

2) Navigare il file system e selezionare il file watch list.

Il set di simboli della watch list viene aggiunto alla finestra *Watch*.

9.2 OSCILLOSCOPIO

L'Oscilloscopio permette di tracciare l'evoluzione dei valori di un set di variabili. Essendo uno strumento asincrono, l'Oscilloscopio non garantisce la sincronizzazione della campionatura.

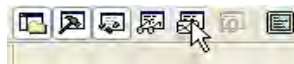
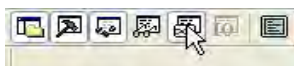
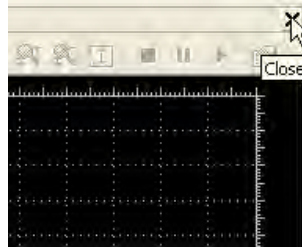
Aprendo l'Oscilloscopio appare una nuova finestra vicino al bordo destro della cornice di LogicLab. Questa è l'interfaccia per avere accesso alle funzioni di debug che l'Oscilloscopio mette a disposizione. L'Oscilloscopio è formato da tre elementi, come mostrato qui sotto.

La barra di applicazione permette di controllare meglio l'Oscilloscopio. Una descrizione dettagliata di ciascun comando verrà data più avanti in questo capitolo.

L'area Chart include numerose voci:

- Plot: area contenente la curva delle variabili.
- Cursori verticali: cursori che identificano due linee verticali distinte. I valori di ciascuna variabile all'intersezione con queste linee sono riportati nelle colonne corrispondenti.
- Barra di scorrimento: se la scala dell'asse x è troppo larga per mostrare tutti i campionamenti nell'area Plot, la barra di scorrimento permette di muoversi avanti e indietro lungo l'asse orizzontale.

La parte più bassa dell'Oscilloscopio è una tabella che presenta una riga per ogni variabile.

9.2.1 APRIRE E CHIUDERE L'OSCILLOSCOPIO

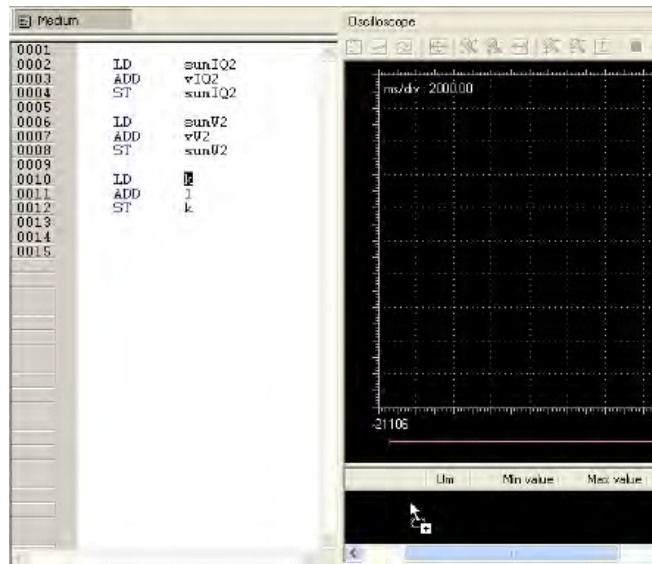
Per aprire l'Oscilloscopio, cliccare sul bottone *Async* sulla barra *Main*.

Per chiudere l'Oscilloscopio, cliccare nuovamente sul bottone *Async*.

Altrimenti, cliccare sul tasto *Close* nell'angolo in alto a destra della finestra dell'Oscilloscopio.

In entrambi i casi, chiudere l'Oscilloscopio significa semplicemente nasconderlo, non resettarlo. A prova di ciò, aprendo nuovamente l'Oscilloscopio dopo averlo chiuso, si vedrà ancora il tracciato della curva delle variabili aggiunte ad esso.

9.2.2 AGGIUNGERE VOCI ALL'OSCILLOSCOPIO

Per tracciare l'evoluzione del valore di una variabile sarà necessario aggiungerla all'Oscilloscopio.

Notare che, a differenza delle finestre trigger e della finestra *Graphic trigger*, è possibile aggiungere all'Oscilloscopio tutte le variabili del progetto, senza preoccuparsi di dove sono dichiarate.

9.2.2.1 AGGIUNGERE UNA VARIABILE DA UN EDITOR DI CODICE SORGENTE TESTUALE

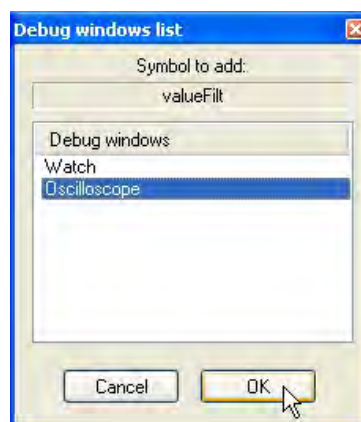
Seguire la procedura seguente per aggiungere una variabile da un editor di codice sorgente testuale (IL o ST) all'Oscilloscopio: selezionare una variabile cliccando due volte su di essa e trascinarla nella finestra *Oscilloscopio*.

La stessa procedura funziona per tutte le variabili che si vogliono inserire.

9.2.2.2 AGGIUNGERE UNA VARIABILE DA UN EDITOR DI CODICE SORGENTE GRAFICO

Seguire la procedura seguente per aggiungere una variabile da un editor di codice sorgente grafico (LD, FBD o SFC) all'Oscilloscopio:

1) Premere il tasto *Watch* nella barra *FBD*.

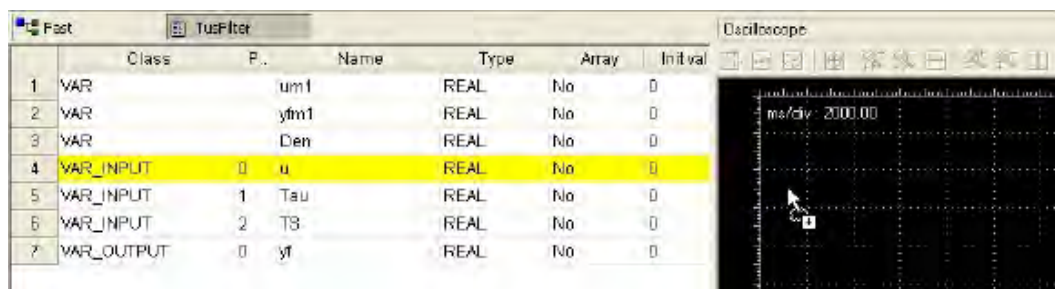
2) Cliccare sul blocco che rappresenta la variabile che si vuole che sia mostrata nell'Oscilloscopio.

3) Apparirà una finestra di dialogo che elenca tutte le ricorrenze di finestre di debug attualmente esistenti, chiedendo quale debba ricevere l'oggetto sul quale si è appena cliccato.

Selezionare *Oscilloscope* e premere *OK*. Il nome della variabile è ora visualizzato nella colonna *Track*.

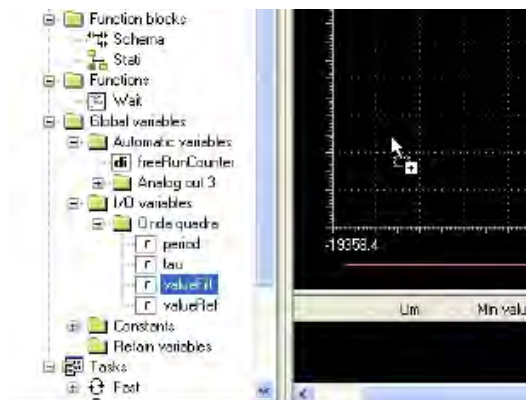
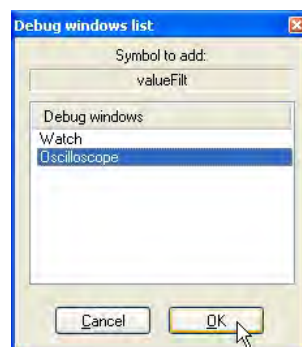
La stessa procedura si applica a tutte le variabili che si vogliono introdurre.

Una volta inserite nell'Oscilloscopio tutte le variabili che si vogliono visualizzare, cliccare sul tasto *Select/Move* della barra *FBD*: il cursore tornerà alla sua forma originale.

9.2.2.3 AGGIUNGERE UNA VARIABILE DA UN EDITOR DI VARIABILI

Per aggiungere una variabile all'Oscilloscopio, selezionare il record corrispondente dell'editor variabili e poi prenderlo e trascinarlo nell'Oscilloscopio

o premere il tasto *F10* e scegliere *Oscilloscope* dalla lista di finestre di debug che compare.

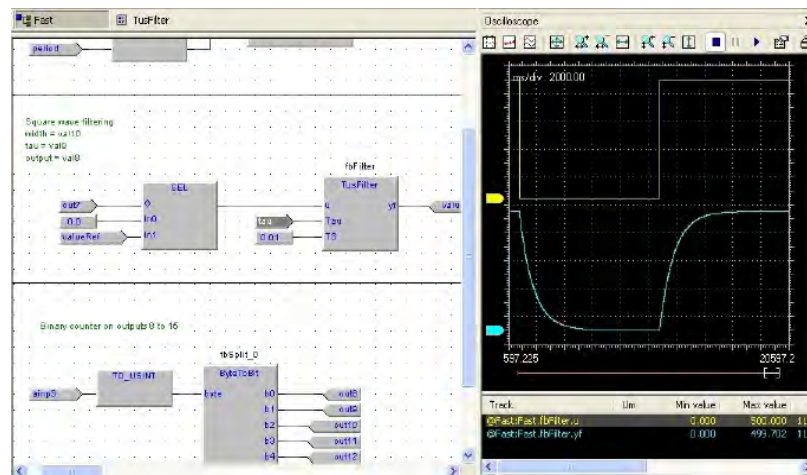
9.2.2.4 AGGIUNGERE UNA VARIABILE DALL'ALBERO DEL PROGETTO

Per aggiungere una variabile all'Oscilloscopio, selezionarla nell'albero del progetto e poi prenderla e trascinarla

o premere il tasto *F10* e scegliere *Oscilloscope* dalla lista di finestre di debug che compare.

9.2.3 ELIMINARE UNA VARIABILE

Per rimuovere una variabile dall'Oscilloscopio, selezionarla cliccando su di essa e premere il tasto *Del*.

9.2.4 VARIABILI DI ESEMPIO

9.2.4.1 OPERAZIONE NORMALE

Considerare l'esempio seguente.

Il manager dell'Oscilloscopio legge periodicamente il valore delle variabili dalla memoria.

Tuttavia, quest'azione avviene in modo asincrono, quindi può succedere che un task a priorità più alta modifichi il valore di alcune delle variabili mentre queste sono lette. Quindi, alla fine del processo di campionamento, i dati associati allo stesso valore dell'asse x potrebbero riferirsi a stati di esecuzione diversi del codice PLC.

9.2.4.2 TARGET DISCONNESSO

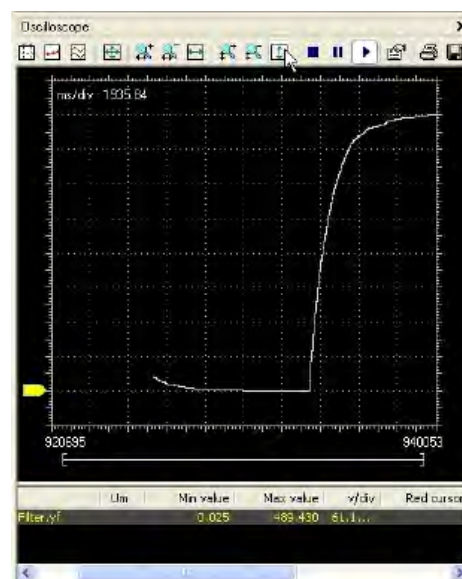
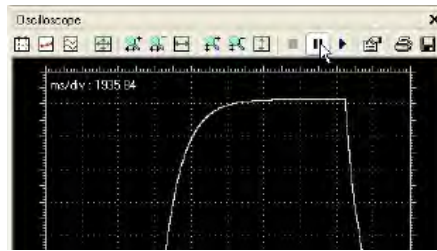
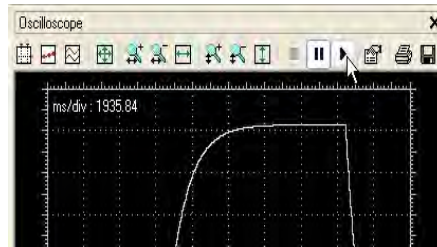
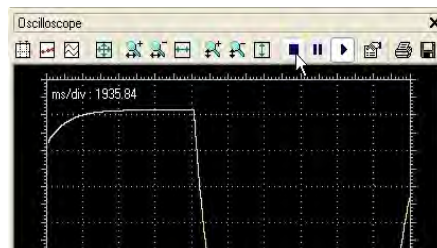
Se il target device è sconnesso, le curve delle variabili coinvolte si bloccheranno fino a quando verrà restaurata la comunicazione.

9.2.5 CONTROLLARE I DATI ACQUISITI E VISUALIZZARLI

L'Oscilloscopio contiene una barra di applicazioni con parecchi comandi, che possono essere usati per controllare il processo di acquisizione e il modo in cui i dati vengono mostrati. Questo paragrafo si focalizza su questi comandi.

Notare che tutti i comandi della barra sono disattivati se non è stata aggiunta nessuna variabile all'Oscilloscopio.

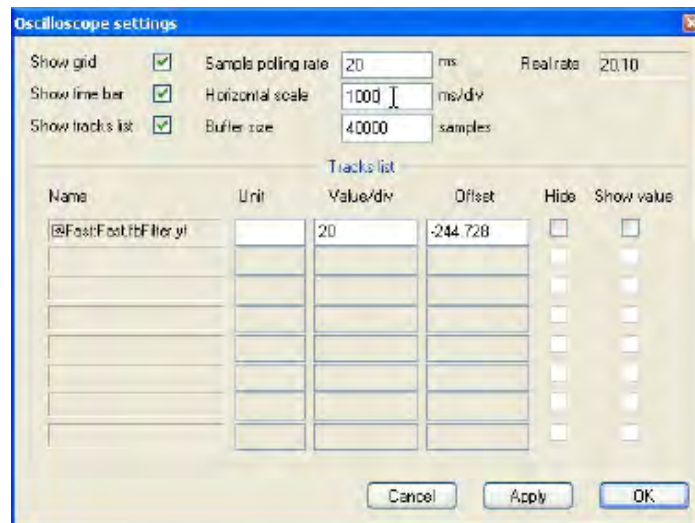
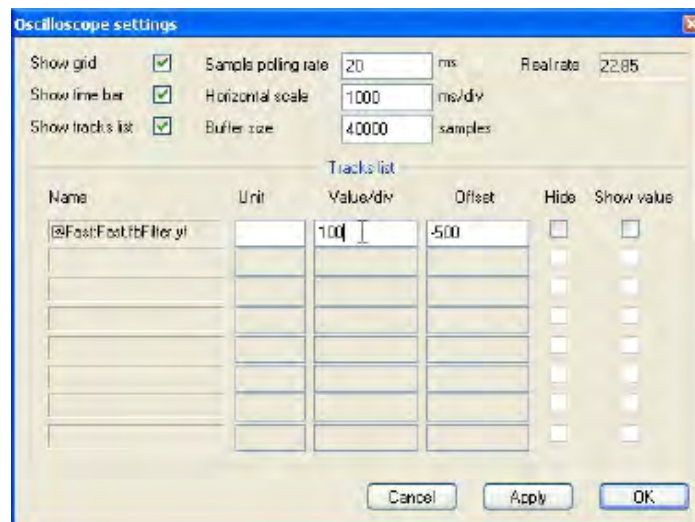
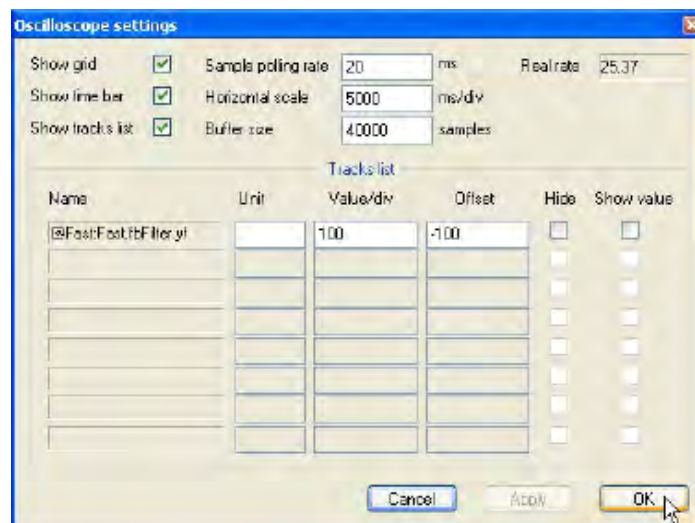
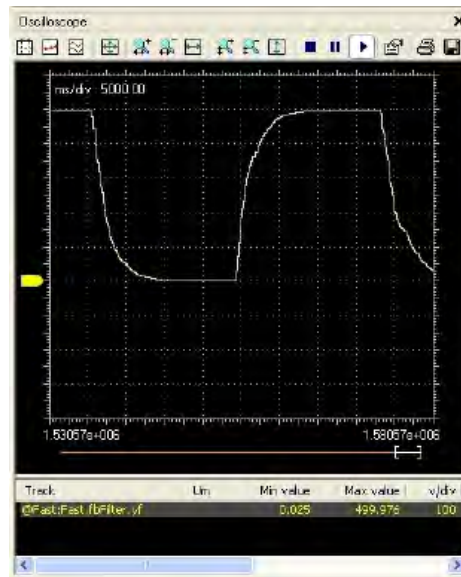
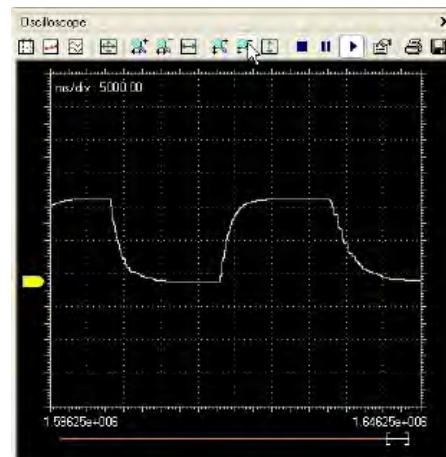
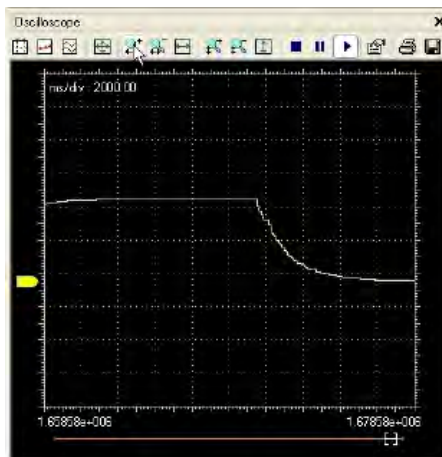
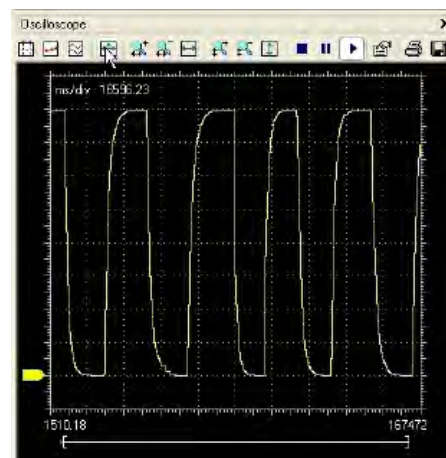
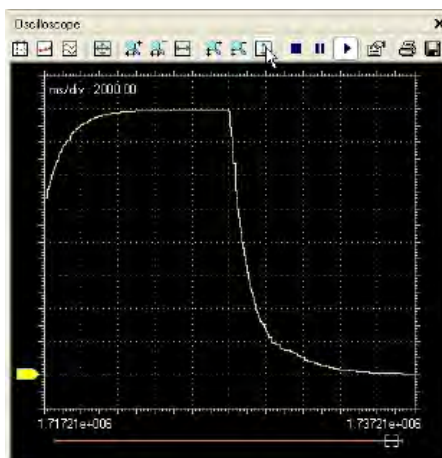
9.2.5.1 INIZIARE E FERMARE L'ACQUISIZIONE DEI DATI

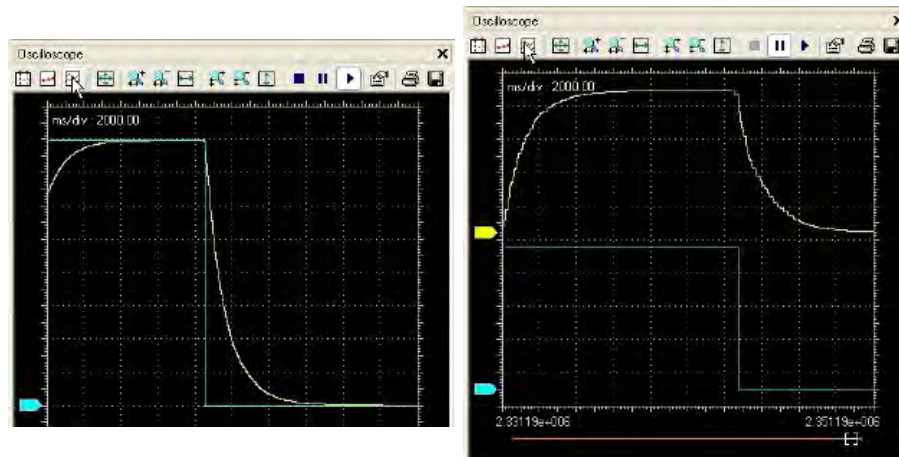
Aggiungendo una variabile all'Oscilloscopio, l'acquisizione di dati parte immediatamente.

Tuttavia, è possibile sospendere l'acquisizione cliccando su `Pause acquisition`.

La curva si blocca (mentre il processo di acquisizione dei dati continua in sottofondo) fino a che non si clicca su `Restart acquisition`.

Per fermare l'acquisizione cliccare su `Stop acquisition`.

In questo caso, cliccando su `Restart acquisition`, l'evoluzione del valore della variabile è tracciato dalla linea di partenza.

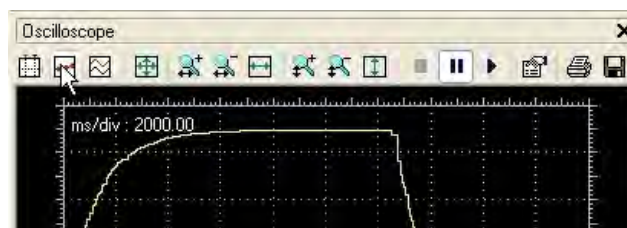
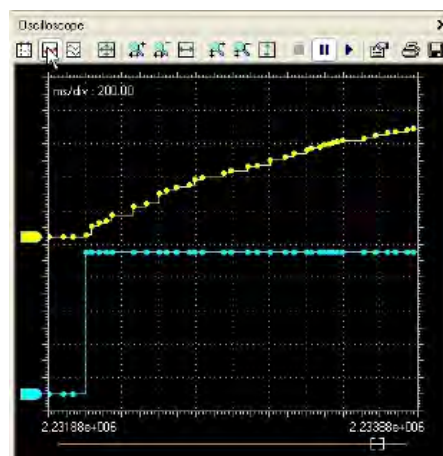
9.2.5.2 IMPOSTARE LA SCALA DEGLI ASSI

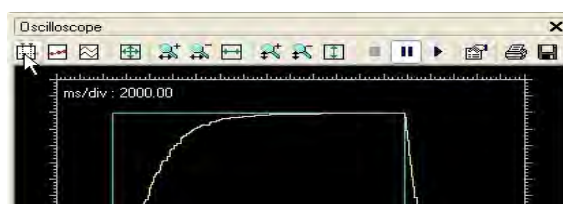
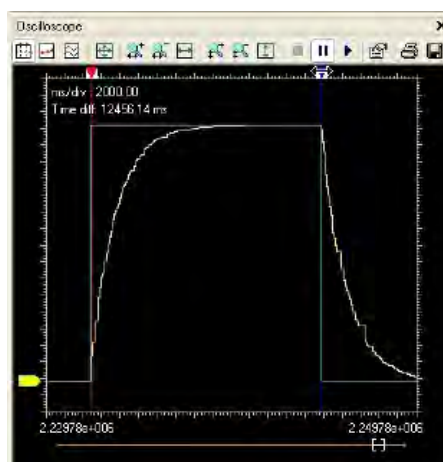
Aprendo Oscilloscopio, LogicLab applica una scala di default agli assi. Tuttavia, se si vuole impostare una scala diversa, seguire questa procedura:

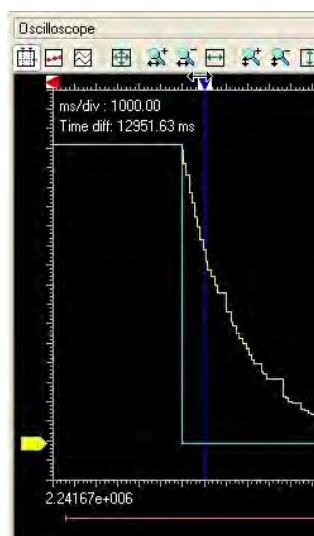
1) Aprire le proprietà del grafico cliccando sulla voce corrispondente della barra.

2) Impostare la scala dell'asse orizzontale, che è comune a tutte le tracce.

3) Per ogni variabile, è possibile specificare una scala distinta per l'asse verticale.

4) Confermare le impostazioni. Il grafico si adatterà per riflettere la nuova scala.

E' inoltre possibile zoommare con rispetto sia dell' asse orizzontale che di quello verticale.

In più, si può adattare velocemente la scala dell'asse orizzontale, di quello verticale o di entrambi per includere tutti i campionamenti, cliccando sulla voce corrispondente della barra.

9.2.5.3 DIVISIONE VERTICALE

Guardando l'evoluzione di due o più variabili, si potrebbe voler dividere le due tracce. Per questo risultato, cliccare sulla voce `Vertical split` nella barra `Oscilloscope`.

9.2.5.4 VISUALIZZARE I SINGOLI CAMPIONI

Cliccando sulla voce `Show samples` nella barra `Oscilloscope`, lo strumento evidenzia i valori singoli riscontrati durante l'acquisizione dei dati.

Si può cliccare ancora sulla stessa voce, per tornare alla modalità di visualizzazione di default.

9.2.5.5 PRENDERE MISURE

L'Oscilloscopio contiene due barre di misurazione, che possono essere sfruttate per prendere alcune misure sul grafico; per mostrarle e nasconderle, cliccare sulla voce *Show measure bars* della barra *Oscilloscope*.

Per misurare un intervallo di tempo tra due eventi, spostare una barra sul punto del grafico che corrisponde al primo evento e l'altra sul punto che corrisponde al secondo.

L'intervallo di tempo tra le due barre è mostrato nell'angolo in alto a sinistra del grafico.

Si può usare una barra di misurazione anche per leggere il valore di tutte le variabili nell'Oscilloscopio in un determinato momento: muovere la barra sul punto del grafico che corrisponde all'istante che si vuole analizzare.

Nella tabella sotto al grafico, si possono leggere i valori di tutte le variabili in un particolare momento.

9.2.5.6 IMPOSTAZIONI DELL'OSCILLOSCOPIO

E' possibile personalizzare ulteriormente l'aspetto dell'Oscilloscopio cliccando sulla voce *Graph properties* nella barra di applicazioni.

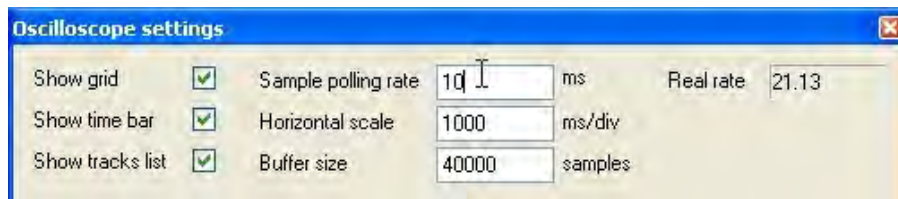
Nella finestra che compare, scegliere se visualizzare *Background grid*, *Time slide bar* e *Track list* oppure no.

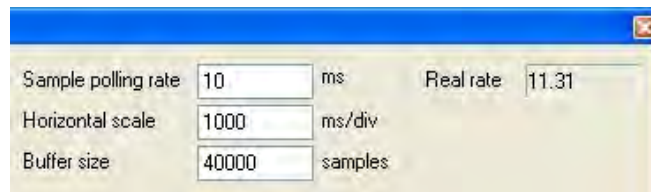
9.2.6 CAMBIARE IL POLLING RATE

LogicLab invia periodicamente richieste al target device, per leggere i dati tracciati nell'Oscilloscopio.

Il polling rate può essere configurato seguendo la procedura seguente:

1) cliccare *Graph properties*.

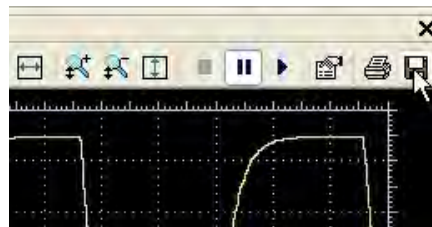
2) Nella finestra che appare modificare *Sampling polling rate*.

3) Confermare l'operazione.

Notare che la rate effettiva dipende dalla performance del target device (in particolare, dalla performance della sua mansione di comunicazione). Si può leggere la rate effettiva nella finestra *Oscilloscope settings*.

9.2.7 SALVARE E STAMPARE IL GRAFICO

LogicLab permette di mantenere l'acquisizione sia salvando i dati su un file sia stampando una visualizzazione dei dati tracciati nell'Oscilloscopio.

9.2.7.1 SALVARE I DATI SU UN FILE

E' possibile salvare i campionamenti acquisiti dall'Oscilloscopio in un file, per poterli ulteriormente analizzare con altri strumenti.

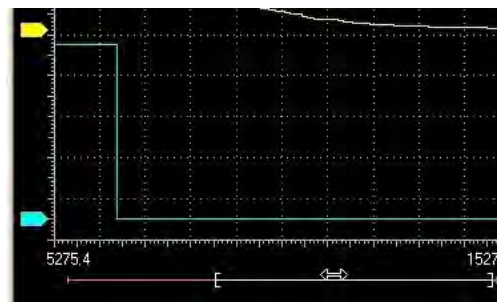
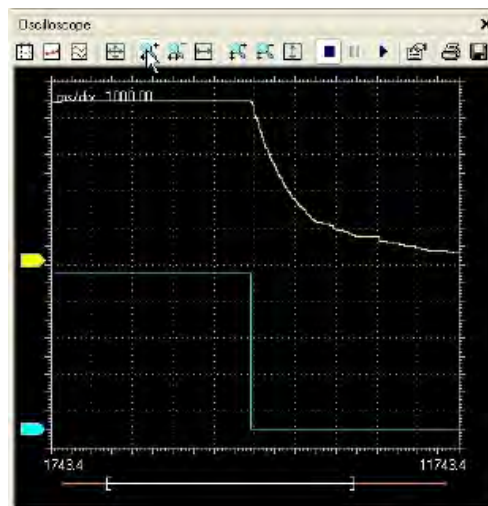
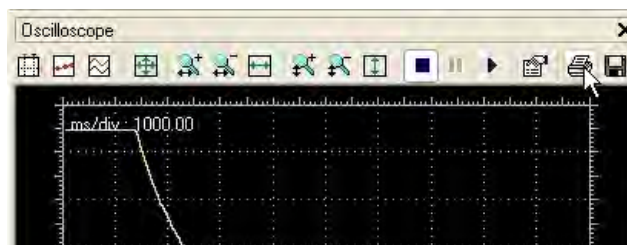
1) Fermare l'acquisizione prima di salvare i dati in un file.
2) Cliccare su *Save tracks data into file* della barra *Oscilloscope*.

3) Scegliere tra i formati di dati disponibili per l'esportazione: OSC è un semplice file di testo, contenente tempo e valore di ogni campione; OSCX è un file XML, che include informazioni più complete, che potranno essere analizzate con altri tool, forniti separatamente da LogicLab.

4) Scegliere il nome del file e la directory di destinazione, poi confermare l'operazione.

9.2.7.2 STAMPARE IL GRAFICO

Seguire questa procedura per stampare una vista dei dati tracciati nell'Oscilloscopio:

1) sospendere o fermare l'acquisizione.

2) Muovere la barra di scorrimento dell'asse dei tempi e sistemare lo zoom, per includere nella visuale gli elementi che si vogliono stampare.

3) Cliccare su *Print graph*.

9.3 MODALITÀ DI MODIFICA E DEBUG

Anche se sia la finestra *Watch* sia l'*Oscilloscope* non utilizzano il codice sorgente, tutti gli altri strumenti di debug sì: quindi, LogicLab richiede allo sviluppatore di passare alla modalità di debug, dove le modifiche al codice sorgente non sono permesse, prima di poter accedere a questi strumenti.

Per attivare e disattivare la modalità debug, cliccare sulla voce corrispondente della barra *Debug*.

Altrimenti, scegliere *Debug mode* nel menu *Project*.

La barra status mostra se la modalità debug è attiva o no.

Notare che non è possibile accedere alla modalità debug se lo status di connessione è diverso da `Connected`.

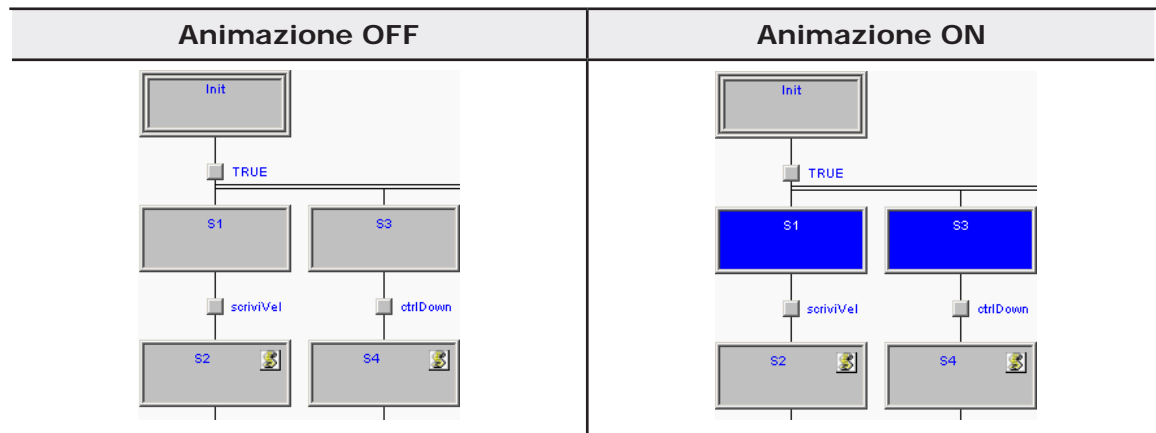
9.4 DEBUG LIVE

LogicLab può mostrare un' animazione esaustiva dell'attuale e variabile stato di esecuzione nel tempo di una POU codificata in qualunque linguaggio di programmazione IEC 61131-3.

Per attivare e disattivare la modalità debug live, cliccare sulla voce corrispondente della barra *Debug*

o scegliere *Live debug mode* dal menu *Project*.

9.4.1 ANIMAZIONE SFC

Come spiegato nella sezione di riferimento del linguaggio, una POU SFC è strutturata come un set di stati, ciascuno dei quali è attivo o inattivo a seconda del momento. Una volta avviato, questo strumento di debug specifico del linguaggio SFC, anima i documenti SFC evidenziando gli stati attivi.

Nella colonna di sinistra, è mostrata una parte di un network SFC, con il diagramma di animazione impostato su off. Nella colonna di destra è mostrata la stessa parte di network quando la modalità debug live è attiva. La figura della colonna di destra mostra che gli stati S1 e S3 sono attivi, mentre Init, S2 e S4 sono inattivi.

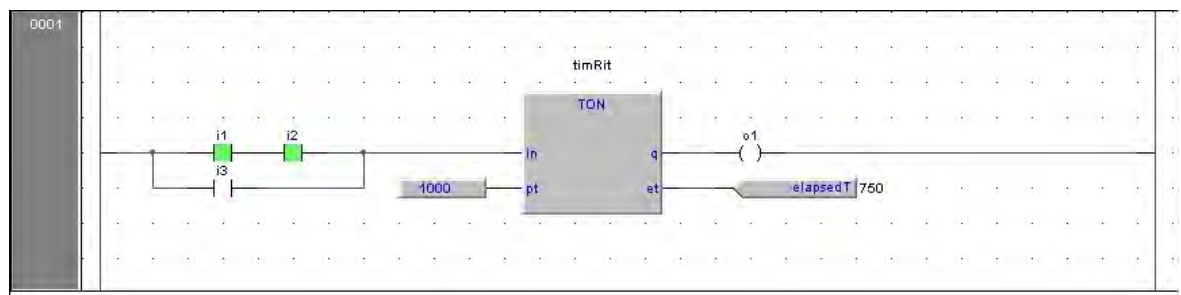
Notare che il manager di animazione SFC testa periodicamente lo stato di tutti gli stati, e l'utente non può modificare il periodo di campionamento. Per questo motivo, può capitare che uno stato rimanga attivo per un periodo di tempo troppo corto da essere visualizzato sul video.

Il fatto che uno stato non sia mai evidenziato non implica che l'azione non venga eseguita, ma potrebbe semplicemente significare che il rate di campionamento è troppo lento per captare l'esecuzione.

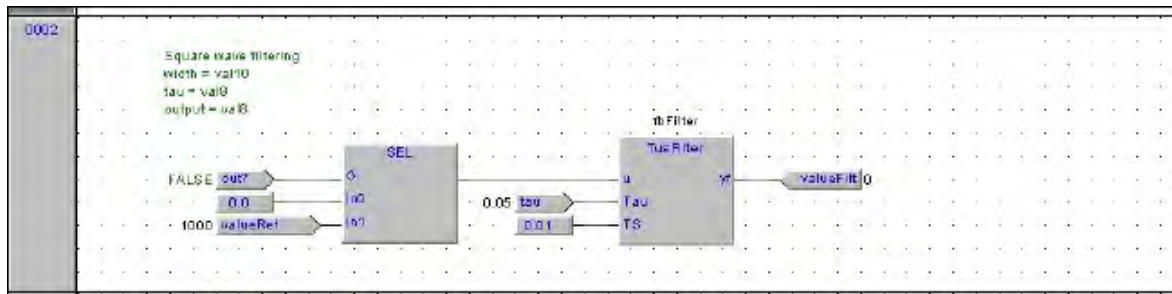
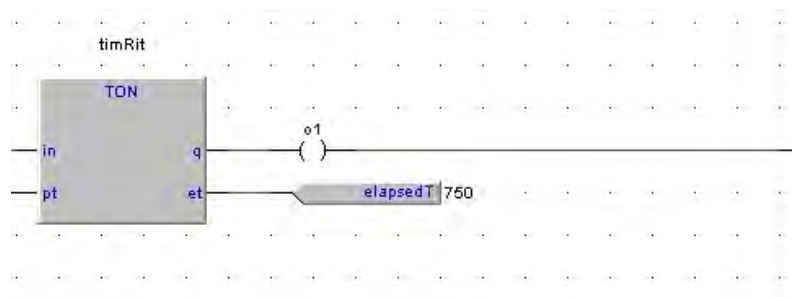
9.4.1.1 DEBUG DI AZIONI E CONDIZIONI

Come spiegato nel riferimento al linguaggio SFC, uno step può essere assegnato ad un'azione, e una transizione ad un codice di condizione. Azioni e condizioni possono essere codificate in uno dei linguaggi IEC 61131-3. Gli strumenti di debug per qualsiasi evenienza possono essere utilizzati all'interno di ciascuna azione/condizione, come se fossero una POU isolata.

9.4.2 ANIMAZIONE LD

Nella modalità debug live, gli schemi Ladder Diagram sono animati evidenziando i contatti e le uscite con valore true (nell'esempio, i1 e i2).

Notare che il manager dell'animazione LD testa periodicamente lo stato di tutti gli elementi. Può capitare che un elemento rimanga true per un periodo di tempo troppo corto da essere visualizzato sul video. Il fatto che un elemento non sia mai evidenziato non implica che il suo valore non diventa mai true (il rate di campionamento potrebbe essere troppo lento).

9.4.3 ANIMAZIONE FBD

Nella modalità debug live, LogicLab mostra i valori di tutte le variabili visibili direttamente nell'editor di codice sorgente grafico.

Questo funziona sia per il linguaggio FBD che per quello LD.

Notare che, anche questo strumento è asincrono.

9.4.4 ANIMAZIONE IL E ST

La modalità debug live si applica anche agli editor di codice sorgente testuale (quelli di IL e ST). Si possono facilmente vedere i valori di una variabile posizionandosi col mouse sopra di essa.

```
0017  (* Analog output 0 = analog inp 0 + analog inp 1 *)
0018
0019  aout0 := ainp0 + ainp1;
0020
0021  (* SFC state logic *)
0022
0023  fbStati( enab := inp10, run := inp11, stop := inp12 );
0024
0025  cnt := cnt + 1;
0026
0027      -29133
```

9.5 TRIGGER

9.5.1 FINESTRA DEI TRIGGER

Lo strumento `Trigger window` permette di selezionare un insieme di variabili e di sincronizzarle in una speciale finestra pop-up.

9.5.1.1 PRE CONDIZIONI PER APRIRE UNA FINESTRA DI TRIGGER

Nessuna necessità di compilazione dedicata

I tool di debug di LogicLab funzionano a run-time. Di conseguenza, a differenza di altri linguaggi di programmazione come il C++, il compilatore non richiede di indicare se supportare o no finestre di trigger: dato un codice PLC, il risultato del compilatore è unico e non vi è distinzione tra versione debug e release.

Memoria disponibile

Una finestra di trigger occupa un segmento di area codice PLC di dimensione ben definita. Ovviamente per far partire una finestra trigger, è necessario che sia disponibile una sufficiente quantità di memoria, altrimenti apparirà un messaggio di errore.

Incompatibilità con le finestre di trigger grafici

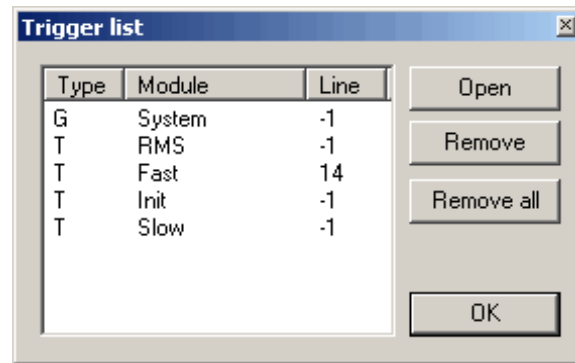
Una finestra di trigger grafico occupa l'intero spazio libero di area codice PLC.

Quindi, una volta che tale strumento di debug è stato avviato, non è possibile aggiungere nessuna finestra di trigger, e un messaggio di errore viene visualizzato se si tenta di avviarene una nuova. Una volta che la finestra di attivazione grafica sarà chiusa, le finestre di trigger saranno di nuovo abilitate.

Si noti che tutte le finestre di trigger esistenti continuano a lavorare normalmente. Non è semplicemente possibile aggiungerne di nuove.

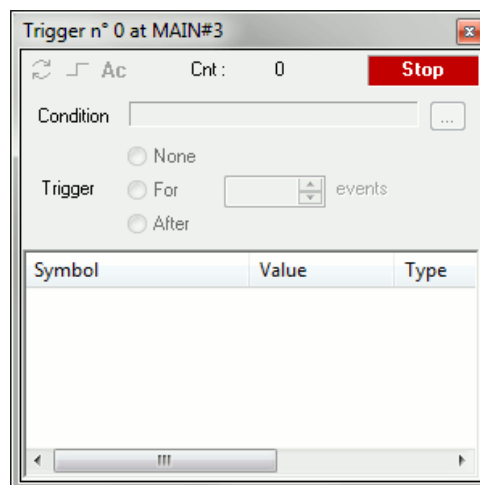
9.5.1.2 TOOLBAR DELLA FINESTRA DEI TRIGGER

L'icona della finestra dei trigger fa parte della barra `Debug` ed è abilitata solo se LogicLab è in modalità debug.

Bottone	Comando	Descrizione
	`Set/Remove trigger`	Per avviare un trigger, selezionare il punto del codice PLC dove inserire il relativo trigger e premere questo bottone. Eseguire la stessa procedura per rimuovere il trigger: per chiudere definitivamente una finestra di debug, cliccare un volta sul blocco istruzione dove è stato inserito il trigger, quindi premere nuovamente questo bottone.
	`Graphic trace`	Questo bottone funziona esattamente come quello sopra `Set/Remove trigger`, salvo che viene aperta una finestra di trigger grafici. Può essere usato allo stesso modo anche per rimuovere un trigger grafico. Tasti shortcut: premere `Shift + F9` è equivalente che premere sul bottone.
	`Remove all triggers`	Premendo questo bottone tutti i trigger ed i trigger grafici esistenti vengono rimossi simultaneamente. Tasti shortcut: premere `Ctrl+Shift+F9` è equivalente che premere sul bottone
	`Trigger list`	Questo bottone apre una finestra con l'elenco di tutte le finestre trigger esistenti. Tasti shortcut: premere `Ctrl+I` è equivalente che premere sul bottone.

Ogni record si riferisce ad una finestre trigger, sia grafico che testuale. La tabella seguente spiega il funzionamento di ciascun campo.

Campo	Descrizione
Type	*T*: finestra trigger. *G*: finestra trigger grafici.
Module	Nome del programma, funzione o blocco funzione dove il trigger è posizionato. Se il modulo è un function block, questo campo contiene il nome del tipo, non dell'istanza per la quale si è installato il trigger.
Line	Per i linguaggi testuali (IL, ST) indica la linea su cui è posizionato trigger. Per gli altri linguaggi il valore è sempre *-1*.

9.5.1.3 INTERFACCIA UTENTE

L'impostazione di un trigger provoca la visualizzazione di una pop-up, questa è l'interfaccia per accedere alle funzioni di debug che la finestra trigger rende disponibili. E' composta da tre elementi, come mostrato qui sotto.

Caption bar

La barra *Caption* della finestra pop-up mostra informazioni sulla posizione del trigger che provoca l'aggiornamento della finestra delle *Variables*, quando è raggiunta dal processore.

Il testo nella barra *Caption* ha il seguente formato:

```
Trigger n° X at ModuleName#Location
```

dove

X	Identificatore di trigger.
ModuleName	Nome del programma, funzione o blocco funzione dove il trigger è posizionato.

Location	Posizione esatta del trigger, all'interno del modulo *ModuleName*.
	Se ModuleName è in IL, *Location* ha il seguente formato.
	`N1`
	Altrimenti, se il *ModuleName* è in FBD, diventa:
	`N2$BT:BID`
	dove:
	`N1` = numero di linea dell'istruzione
	`N2` = numero network
	`BT` = tipo di blocco (operando, funzione, blocco funzione, etc.)
	`BID` = identificazione del blocco

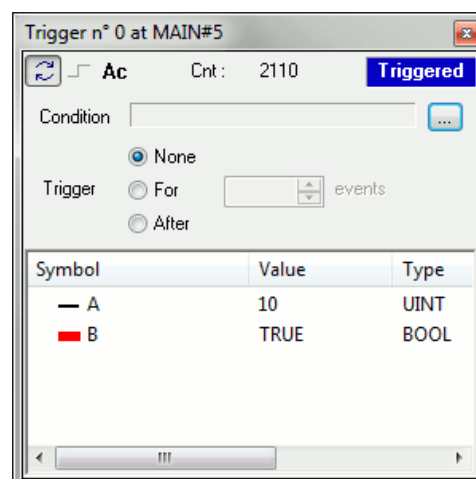
Sezione controlli

Questa finestra di dialogo consente all'utente di controllare meglio il refresh della finestra di trigger per ottenere maggiori informazioni sul codice. Una descrizione dettagliata delle funzioni di ogni controllo è data nella sezione controlli *Trigger window* (vedi 9.5.2.11).

In tutti i comandi tranne *Ac*, il bottone *Accumulator display*, non è accessibile fino a quando almeno una variabile è trascinata dentro la finestra debug.

Sezione variabili

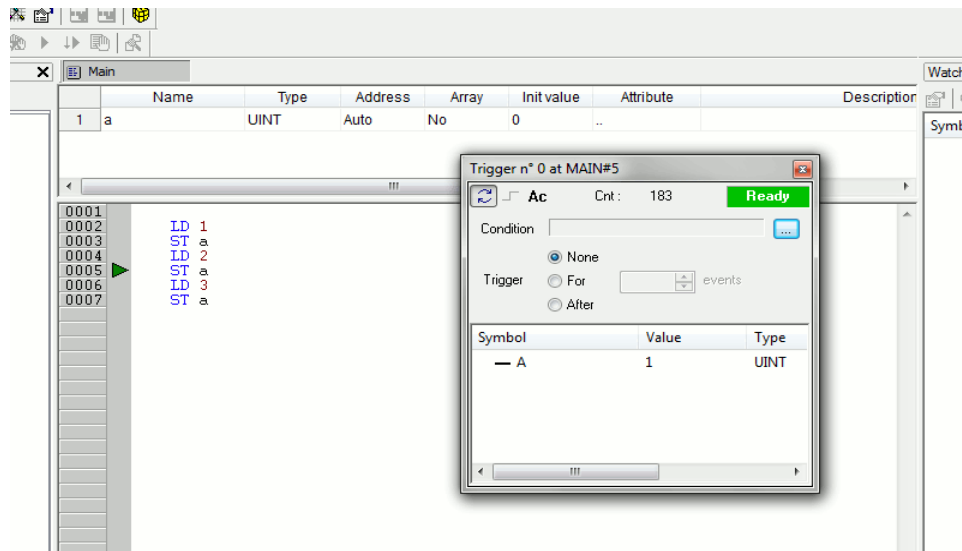
La parte più bassa della finestra *Debug* è una tabella composta da una riga per ogni variabile trascinata dentro.

Ogni riga ha quattro campi: il nome della variabile, il suo valore, il tipo, e la posizione (`@ task:ModuleName`) letti dalla memoria durante l'ultimo aggiornamento.

9.5.1.4 FINESTRA TRIGGER: INFORMAZIONE SU DRAG AND DROP

Per vedere una variabile, è necessario copiarla nella parte bassa della finestra *Debug*.

Questa sezione è una tabella composta da una riga per ogni variabile trascinata dentro. Potete trascinare dentro la finestra trigger solo variabili locali al modulo dove avete posizionato il relativo trigger, o variabile globale o parametro. Non potete trascinare una variabile dichiarata in un altro programma, funzione o blocco funzione.

9.5.1.5 AGGIORNAMENTO DEI VALORI

Si consideri il seguente esempio.

Il valore delle variabili viene aggiornato ogni volta che viene attivato il window manager, ogni volta che il processore esegue le istruzioni segnate da una freccia verde. Tuttavia è possibile impostare i controlli in modo da ottenere l'aggiornamento delle variabili solo quando i trigger soddisfano le condizioni definite.

Si noti che il valore della variabile nella colonna *Symbol* è letto dalla memoria appena prima dell'istruzione marcata con il simbolo di trigger (in questo caso: l'istruzione alla linea 5) e immediatamente dopo l'istruzione precedente (linea 4) a cui è stata assegnata.

Dunque, nell'esempio precedente la seconda istruzione ST non è stata ancora eseguita quando il nuovo valore di *a* viene letto dalla memoria per essere visualizzato nella finestra di trigger.

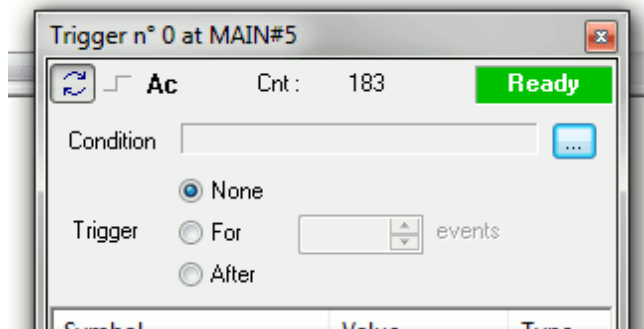
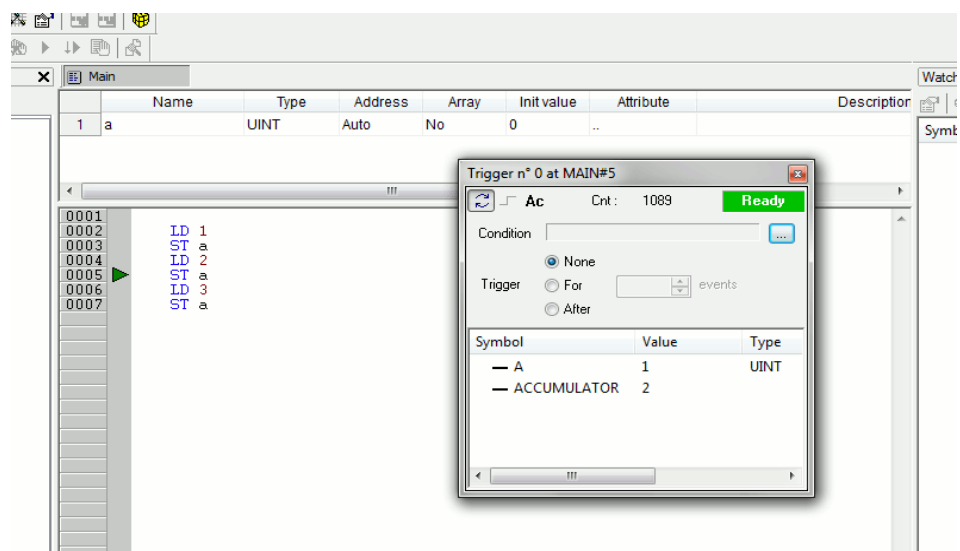
Perciò il valore di *a* è 1.

9.5.1.6 FINESTRA DEI CONTROLLI

Questo paragrafo tratta i controlli della finestra trigger, che permette di avere una visione migliore del lavoro di questo strumento di debug, per avere maggiori informazioni sul codice.

I controlli della finestra trigger agiscono in modo definito sul comportamento della finestra, indipendentemente dal tipo di modulo (IL o FBD) dove il trigger è stato inserito.

Tutti i controlli ad eccezione di *Accumulator display* non sono accessibili fino a quando almeno una variabile è trascinata dentro la finestra delle *Variables*.

I controlli della finestra sono resi accessibili all'utente attraverso la parte grigia superiore della finestra di debug.

Bottoni	Comando	Descrizione
	Start/Stop	Questo controllo è usato per avviare una sessione di trigger. Se il trigger è avviato sul sistema potete premere questo bottone per forzare l'arresto. In caso contrario la sessione viene automaticamente fermata quando viene raggiunta una condizione. A questo punto potete premere questo bottone per avviare un'altra sessione di trigger.
	Single step execution	Questo controllo è usato per eseguire un singolo step del trigger. E' abilitato solo quando non ci sono sessioni attive del trigger e *None* è selezionato. Sono considerate condizioni specifiche. Finita l'esecuzione del singolo step del trigger, la sessione viene automaticamente stoppata.
	Accumulator display	Questo controllo aggiunge l' *Accumulator* alla lista delle variabili già trascinate dentro la finestra trigger. Viene aggiunta una nuova riga alla fine della tabella delle variabili, contenente la stringa *Accumulator* nella colonna *Symbol*, il valore dell'accumulatore nella colonna *Value*, *Type* non è specificato e *Location* è settato a livello globale come mostrato nella figura seguente.

Per rimuovere l'accumulatore dalla tabella, cliccare sul suo nome nella colonna *Symbol*, e premere il tasto *Del*.

Questo controllo può essere molto utile se un trigger è stato inserito in una dichiarazione ST, perché permette di conoscere quale valore viene scritto nella variabile di destinazione, durante l'esecuzione corrente del task. E' possibile ottenere lo stesso risultato spostando il trigger all'istruzione successiva segnata da una freccia verde.

Contatore dei Trigger

Questo controllo di sola lettura conta quante volte il gestore della finestra debug è stata attivata, da quando la finestra è stata installata.

Il gestore delle finestre ripristina automaticamente questo contatore tutte le volte che una nuova sessione di trigger viene attivata.

Stato dei Trigger

Questo controllo di sola lettura mostra lo stato della finestra di *Debug*. Può assumere i seguenti valori.

Ready	Il trigger non si è verificato durante la corrente esecuzione del task.
Triggered	Il trigger si è verificato durante la corrente esecuzione del task.
Stop	Il sistema non sta eseguendo il trigger. Il trigger non è stato fatto partire oppure è stato fermato dall'utente oppure si è verificata una condizione di stop.
Error	La comunicazione con il target è stata interrotta, lo stato della finestra trigger non può essere determinato.

Condizioni definite dall'utente

Se viene definita una condizione usando questo controllo, il valore nella finestra *Debug* viene aggiornato tutte le volte che il gestore delle finestre viene attivato e le condizioni definite dell'utente sono vere.

Dopo avere inserito una condizione, il controllo visualizza la sua espressione semplificata.

Contatori

Questo controllo permette di definire condizioni sul contatore dei trigger.

La finestra trigger può essere in uno dei seguenti tre stati.

- *None*: non è stato avviato nessun contatore, quindi non sono state specificate condizioni sul trigger.

- *For*: partendo al presupposto che avete dato il valore `N` come limite del contatore, il gestore delle finestre aggiunge *1* al valore corrente del contatore e aggiorna il valore di questa variabile ogni volta che la finestra debug è attivata. Tuttavia, quando il contatore è uguale ad `N`, la finestra smette di aggiornare i valori, e cambia lo stato visualizzando *Stop*.
- *After*: partendo al presupposto che avete dato il valore `N` come limite del contatore, il gestore delle finestre fa ripartire il contatore e aggiunge *1* al suo valore corrente ogni volta che è attivato. La finestra rimane sullo stato *Ready* e non aggiorna il valore delle sue variabili fino a quando il contatore raggiunge `N`.

9.5.2 DEBUG CON LA FINESTRA TIRGGER

9.5.2.1 INTRODUZIONE

Lo strumento finestra trigger permette all'utente di selezionare un insieme di variabili e di avere il loro valore visualizzato e aggiornato in modo sincrono su una finestra pop-up. A differenza della finestra *Watch*, la finestra trigger aggiorna simultaneamente tutte le variabili che contiene, ogni volta che sono attivate.

9.5.2.2 APRIRE UNA FINESTRA TRIGGER DA UN MODULO IL

Supponiamo di essere in un modulo IL, che contenga le seguenti istruzioni.

```
0001
0002  LD   a
0003  ADD  b
0004  ST   a
0005
0006  LD   c
0007  ADD  d
0008  ST   c
0009
0010  LD   k
0011  ADD  1
0012  ST   k
0013
```

Supponiamo anche che volete conoscere il valore di `b`, `d` e `k`, appena prima che venga eseguita l'istruzione ST *k*. Per fare questo spostare il cursore sulla riga 12.

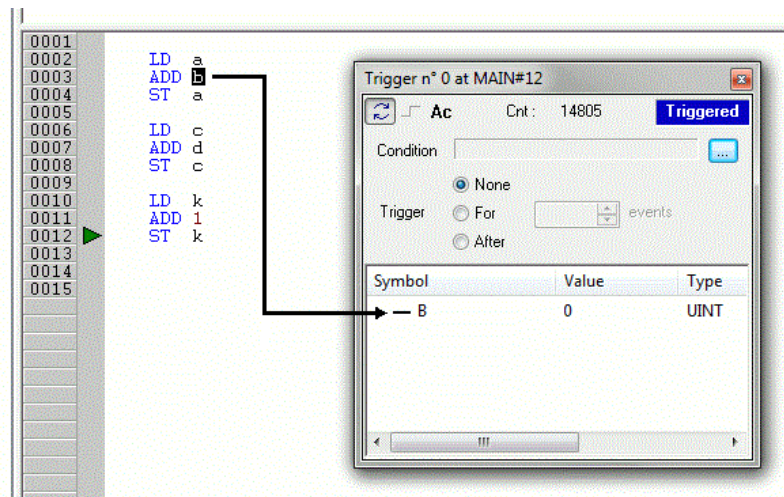
```
0009
0010  LD   k
0011  ADD  1
0012  ST   k
0013
```

Quindi è possibile fare click sul bottone *Set/Remove trigger* sulla barra *Debug*.

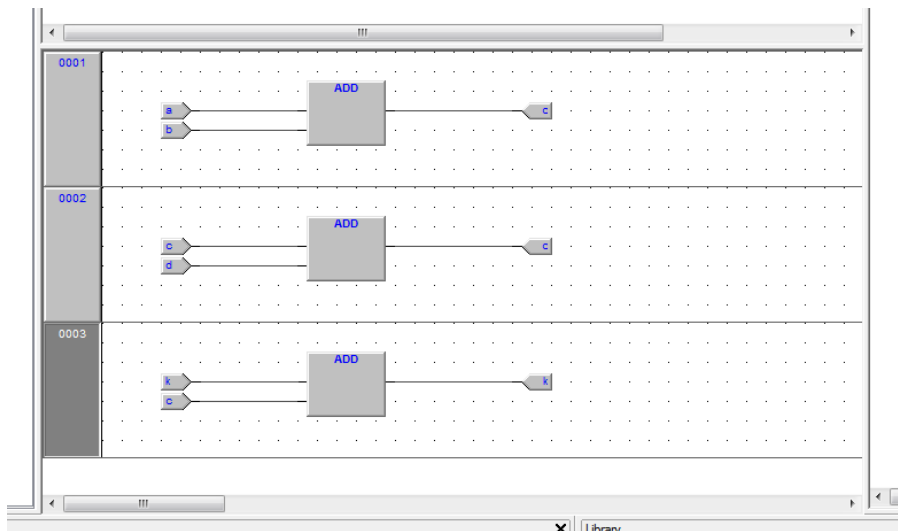
o premere il tasto *F9*.

In ogni caso, una freccia verde appare vicino al numero della linea, e viene aperta la relativa finestra trigger pop-up. Non tutte le istruzioni IL supportano i trigger. Per esempio, non è possibile inserire un trigger all'inizio di una riga che contiene un'istruzione `JMP`.

9.5.2.3 AGGIUNGERE UNA VARIABILE AD UNA FINESTRA TRIGGER DA UN MODULO IL

Per vedere il valore di una variabile, è necessario aggiungerla alla finestra trigger. Per fare questo, selezionare una variabile fare doppio click, e trascinarla dentro la finestra delle *Variables*, il rettangolo in basso bianco della finestra pop-up. Il nome della variabile appare ora nella colonna *Symbol*.

Applicare la stessa procedura per tutte le variabili che si vogliono controllare.

9.5.2.4 APRIRE UNA FINESTRA TRIGGER DA UN MODULO FBD

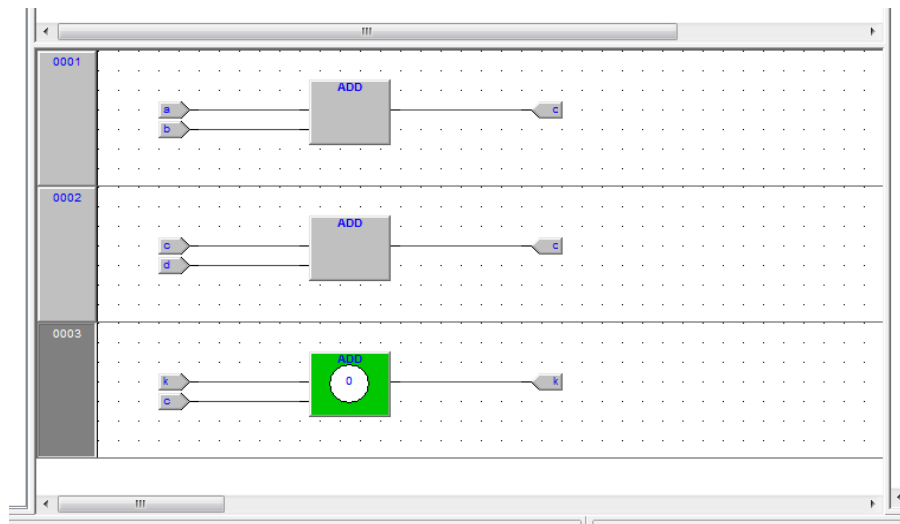
Supponiamo di avere un modulo FBD, che contenga le seguenti istruzioni.

Supponiamo anche di volere conoscere il valore di `C`, `D` e `K`, appena prima che venga eseguita l’istruzione ST k.

Sapendo che non si può mai inserire un trigger in un blocco che rappresenta una variabile come

potete selezionare il primo blocco disponibile precedente la variabile selezionata. Nell’esempio della figura sopra, potete spostare il cursore al network 3 e premere il blocco `ADD`.

Potete premere il bottone *Set/Remove trigger* sulla barra *Debug*

o premere il tasto *F9*.

In ogni caso, il colore del blocco selezionato diventa grigio, in mezzo al blocco appare un cerchio bianco con un numero dentro e la relativa finestra trigger pop-up appare.

All'atto di preprocessare il codice sorgente FBD, il compilatore lo traduce in istruzioni di linguaggio IL. L'istruzione *ADD* al network 3 è esteso a:

```
LD  k
ADD 1
ST  k
```

Quando aggiungete un trigger in un blocco FBD, dovete posizionare il trigger prima della prima istruzione del suo codice IL equivalente.

9.5.2.5 AGGIUNGERE UNA VARIABILE AD UNA FINESTRA TRIGGER DA UN MODULO FBD

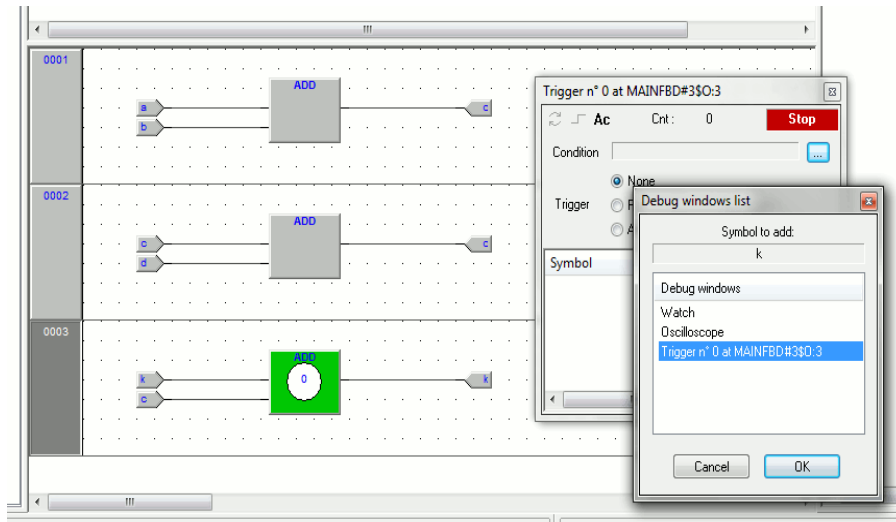
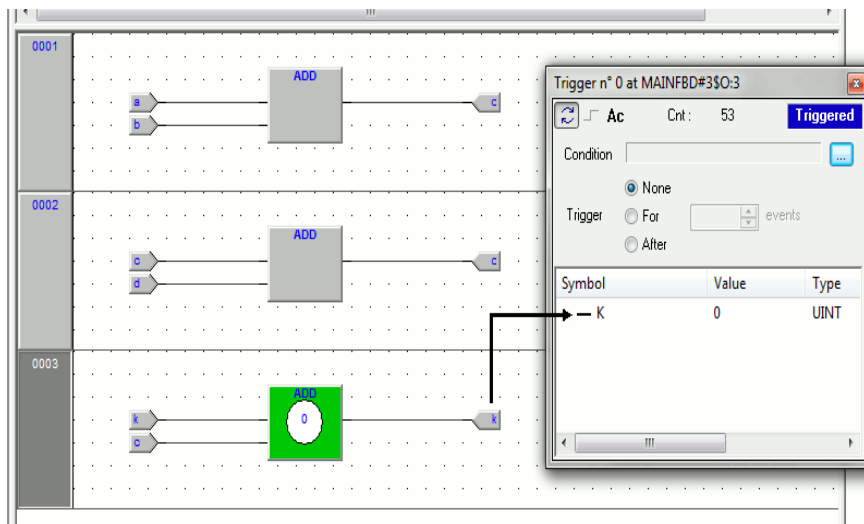
Per vedere il valore di una variabile, bisogna aggiungerla alla finestra trigger. Supponiamo che si vuole controllare il valore della variabile k del codice FBD nella figura sottostante.

A questo proposito, premere il bottone *Watch* nella barra FBD.

Il cursore diventerà come il seguente.

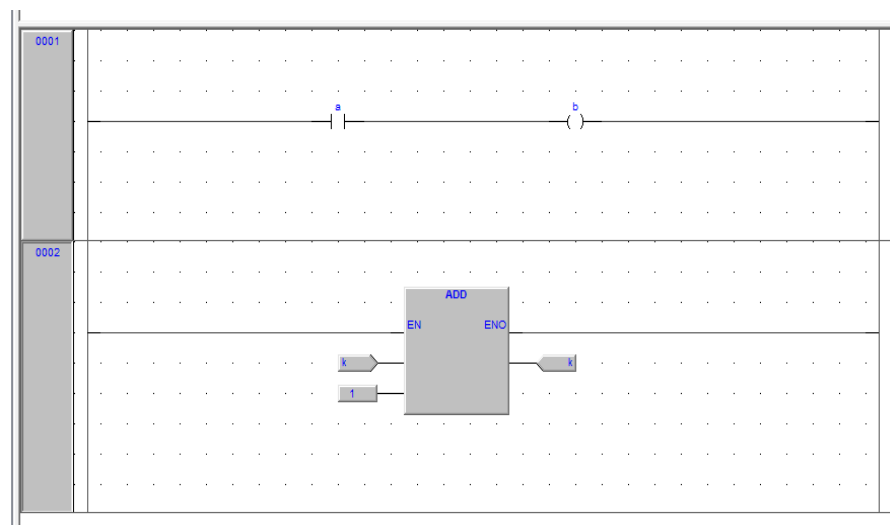
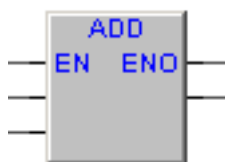
Ora è possibile fare clic sul blocco che rappresenta la variabile che desiderate venga visualizzata nella finestra trigger.

Nell'esempio che stiamo considerando, cliccare il blocco.

Appare una finestra di dialogo con l'elenco di tutte le istanze correnti esistenti della finestra debug, e chiede quale deve ricevere l'oggetto che avete appena cliccato.

Al fine di visualizzare la variabile k nella finestra trigger, selezionare il suo riferimento nella colonna `Debug windows` poi premere `OK`. Il nome della variabile è ora stampato nella colonna `Symbol`.

Applicare la stessa procedura per tutte le variabili che si vogliono controllare.

Una volta aggiunte alla finestra `Graphic watch` tutte le variabili che si vogliono osservare, potete premere il bottone con il cursore normale, così il cursore riprenderà la sua forma originale.

9.5.2.6 APRIRE UNA FINESTRA TRIGGER DA UN MODULO LD

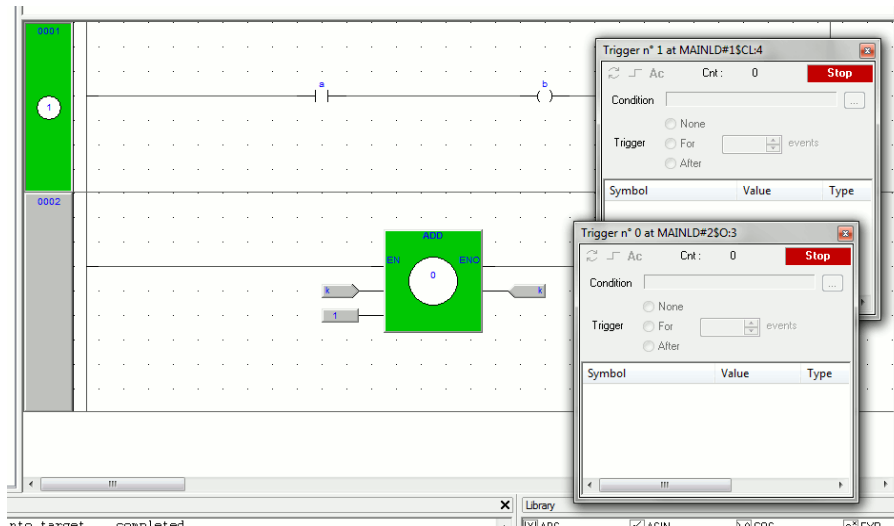
Supponiamo di avere un modulo LD, che contenga le seguenti istruzioni.

Potete inserire un trigger su un blocco come il seguente.

In questo caso, vengono applicate le stesse regole usate per inserire un trigger in un modulo FBD su un contatto

o un'uscita.

In questo caso, seguire le istruzioni del linguaggio ST. Supponiamo di voler conoscere il valore di alcune variabili tutte le volte che il processore raggiunge il network numero 1.

Per prima cosa premere uno degli oggetti che compongono il network numero 1. Ora potete cliccare sul bottone *Set/Remove trigger* della barra *Debug*.

In alternativa potete premere sul tasto *F9*.

In ogni caso, il pulsante grigio in rilievo contenente il numero del network ritorna verde e un cerchio bianco con il numero del trigger appare nel mezzo del bottone, mentre appare la relativa pop-up.

A differenza degli altri linguaggi supportati da LogicLab, LD non permette di inserire un trigger in un singolo contatto o in un'uscita, in quanto permette di selezionare solo un network intero. Quindi le variabili nella finestra trigger saranno aggiornate tutte le volte che il processore raggiungerà l'inizio del network selezionato.

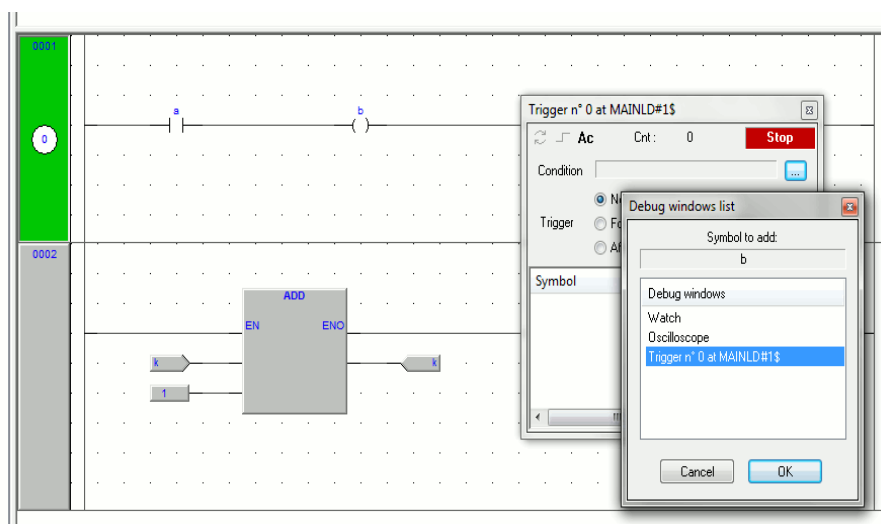
9.5.2.7 AGGIUNGERE UNA VARIABILE AD UNA FINESTRA TRIGGER DA UN MODULO LD

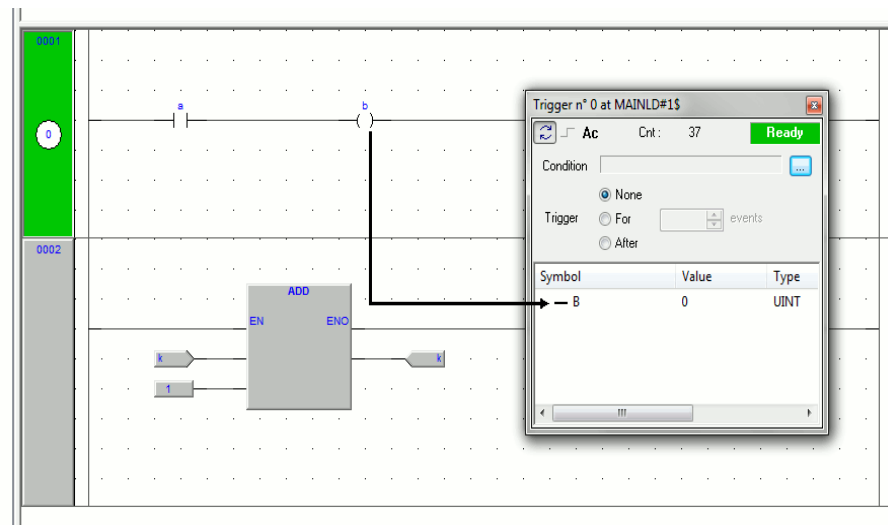
Per vedere il valore di una variabile, è necessario aggiungerla nella finestra trigger. Supponiamo di voler controllare il valore della variabile b nel codice LD rappresentato nella figura seguente.

A questo proposito, premere il bottone *Watch* nella barra *FBD*.

Il cursore diventa come il seguente.

Ora potete premere l'oggetto che rappresenta la variabile che volete mostrare nella finestra trigger.

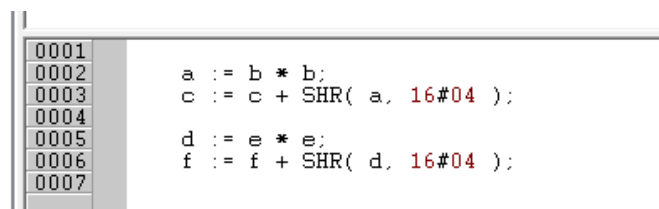
Appare una finestra di dialogo con l'elenco di tutte le istanze correnti esistenti nella finestra debug e chiede quale deve ricevere l'oggetto che avete appena cliccato.

Per visualizzare la variabile `B` nella finestra trigger, selezionare il suo riferimento nella colonna *Debug window* e premere *OK.*

Il nome della variabile è ora stampata nella colonna *Symbol.*

Applicare la stessa procedura a tutte le variabili che si vogliono controllare.

Una volta aggiunte alla finestra *Graphic watch* tutte le variabili che si vogliono osservare, potete premere il bottone *Normal cursor*, così si ripristina la forma originale del cursore.

9.5.2.8 APRIRE UNA FINESTRA TRIGGER DA UN MODULO ST

Supponiamo di avere un modulo ST, che contenga le seguenti istruzioni.

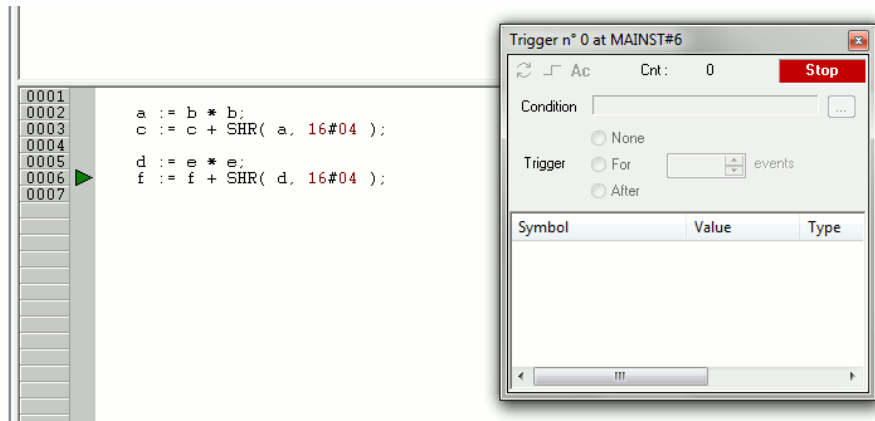
```
0001
0002    a := b * b;
0003    c := c + SHR( a, 16#04 );
0004
0005    d := e * e;
0006    f := f + SHR( d, 16#04 );
0007
```

Supponiamo inoltre di voler conoscere il valore di `e`, `d` e `f`, appena prima che venga eseguita l'istruzione

```
f := f+ SHR( d, 16#04 )
```

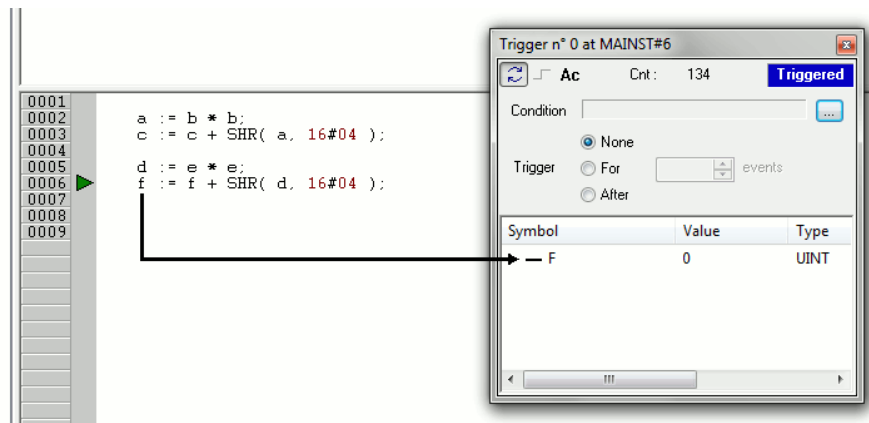
Per fare questo spostare il cursore alla linea 6.

Quindi premere il pulsante *Set/Remove trigger* sulla barra *Debug.*

o premere il tasto *F9.*

In entrambi i casi, apparirà una freccia verde vicino al numero della linea e si aprirà la relativa finestra pop-up.

Non tutte le istruzioni ST supportano i trigger. Per esempio, non è possibile posizionare un trigger su una linea che contiene una terminazione come END_IF, END_FOR, END_WHILE, etc..

9.5.2.9 AGGIUNGERE UNA VARIABILE AD UNA FINESTRA TRIGGER DA UN MODULO ST

Per vedere il valore delle variabili, è necessario aggiungerla in una finestra trigger. A questo proposito, selezionare una variabile, facendo doppio click su di essa, quindi trascinarla dentro la finestra delle *Variables*, nella parte bianca in basso nella finestra pop-up. Il nome della variabile ora appare nella colonna *Symbol*.

Applicare la stessa procedura a tutte le variabili che si vogliono controllare.

9.5.2.10 RIMUOVERE UNA VARIABILE DALLA FINESTRA TRIGGER

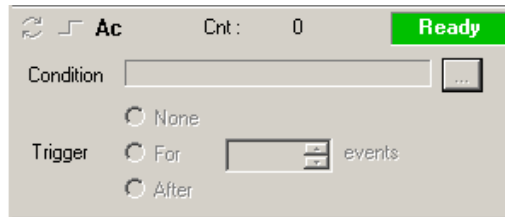
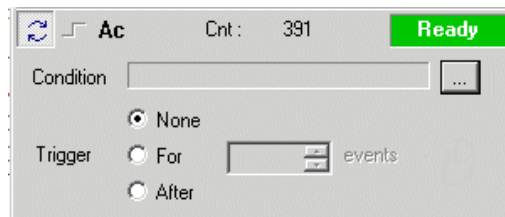
Se si desidera che una variabile non venga visualizzata più nella finestra trigger, selezionarla, cliccando sul suo nome una volta, quindi premere il tasto *Del*.

9.5.2.11 USARE I CONTROLLI

Questo paragrafo tratta i controlli della finestra trigger, che consentono di visionare meglio il funzionamento dello strumento debug per avere maggiori informazioni sul codice interessato.

Lo scopo principale dei controlli della finestra trigger è che consente di definire maggiori limiti sui controlli, così che le variabili nella finestra *Variables* vengano aggiornate quando il processore raggiunge la posizione del trigger e questa condizione è soddisfatta.

Abilitare i controlli

Quando si imposta un trigger, tutti gli elementi nella finestra *Control* sono disabilitati.

Infatti, non è possibile accedere ai controlli, ad eccezione del display *Accumulator*, fino a quando almeno una variabile verrà trascinata nella finestra *Debug*. Quando questo accadrà verrà avviato automaticamente il trigger e la finestra *Controls* cambierà, come mostrato nell'immagine seguente.

I trigger possono essere avviati o stoppati con l'apposito bottone.

Fissare il numero degli aggiornamenti

Se volete aggiornare le variabili la prima volta che la finestra sia avviata, selezionare *None*, e premere il bottone step singolo, in caso contrario impostare il contatore ad *1* e selezionare *For*.

Se volete aggiornare le variabili per un numero di campioni *X* a partire dall'attivazione della condizione di trigger, impostare il contatore ad *X* e selezionare *For*.

Se volete che i valori siano aggiornati solo dopo un numero di campioni *Y* a partire dall'attivazione della condizione di trigger, impostare il contatore ad *Y* e selezionare *After*.

Le impostazioni relative ai trigger e alle loro condizioni diventano attive quando il trigger viene fatto ripartire.

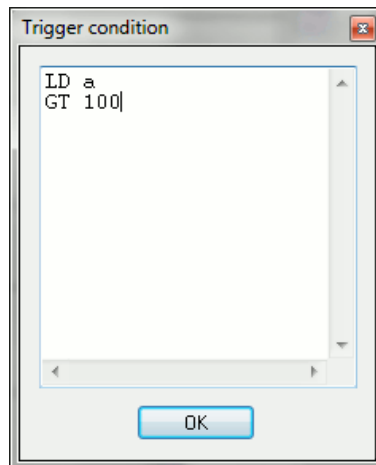
Guardare l'accumulatore

Come indicato nel capitolo Aggiornamento dei valori (vedi 9.5.1.5), quando si inserisce un trigger su una linea di istruzione, si stabilisce che le variabili nella relativa finestra di debug saranno aggiornate ogni volta che il processore raggiungerà questa posizione, prima che l'istruzione venga eseguita. In alcuni casi, per esempio quando il trigger è posizionato prima di un'istruzione ST, può essere utile conoscere il valore dell'accumulatore. Ciò consente di prevedere l'esito dell'istruzione che verrà eseguita dopo che tutte le variabili nella finestra trigger saranno state aggiornate. Per aggiungere l'accumulatore alla finestra dei trigger, cliccare sul bottone *Accumulator display*.

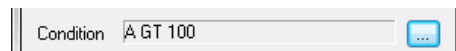
Definire una condition

Questo controllo consente all'utente di impostare una condizione sulle occorrenze di un trigger. Di default, questa condizione è impostata a *TRUE* e i valori nella finestra debug sono aggiornati tutte le volte che il gestore della finestra è attivato.

Se volete una restrizione sul meccanismo degli aggiornamenti, potete specificare una condizione cliccando sull'apposito bottone.

Premendo il bottone si apre una finestra di testo, dove poter scrivere il codice IL che imposta la condizione.

Una volta scritta la condizione, premere il bottone *OK* per istallarla o premere il bottone *Esc* per cancellare l'operazione.

Se si sceglie di istallarla, il valore nella finestra debug è aggiornato tutte le volte che si attiva il gestore della finestra e le condizioni definite dall'utente sono vere.

Una espressione semplificata della condizione appare ora nel controllo.

Per modificarla, premere nuovamente sul bottone indicato.

Appare la finestra di testo, contenente il testo scritto precedentemente e che potete ora modificare.

Per rimuovere completamente una condizione definita dall'utente, eliminare l'intero codice IL nella finestra di testo e premere *OK*.

Dopo l'esecuzione della condizione, l'accumulatore deve essere di tipo Booleano (*TRUE* o *FALSE*), altrimenti si verificherà un errore di compilazione.

Solo le variabili globali e le variabili trascinate all'interno della finestra possono essere usate per le condizioni.

Ovvero, tutte le variabili locali al modulo dove il trigger è stato originariamente inserito sono fuori portata, se non sono state trascinate dentro la finestra debug. Nella finestra delle condizioni non possono essere dichiarate nuove variabili.

9.5.2.12 CHIUDERE LA FINESTRA TRIGGER E RIMUOVERE UN TRIGGER

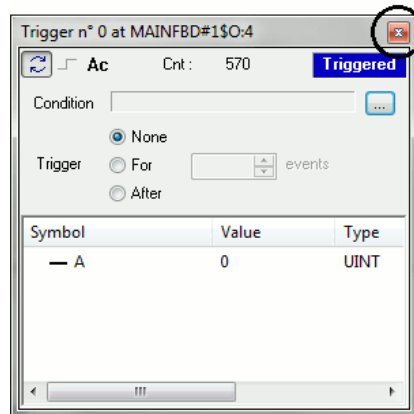
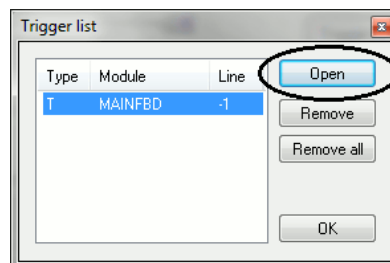
Questa pagina web si occupa di cosa si può fare quando si termina una sessione di debug con una finestra di trigger. È possibile scegliere tra le seguenti opzioni.

- Chiudere la finestra dei trigger.
- Rimuovere il trigger.
- Rimuovere tutti i trigger.

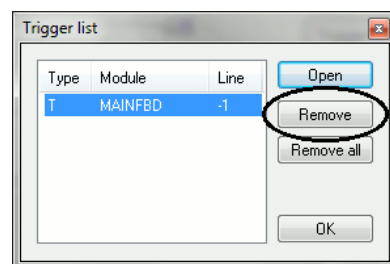
Notare che le azioni elencate sopra producono risultati molto diversi.

Chiudere la finestra dei trigger

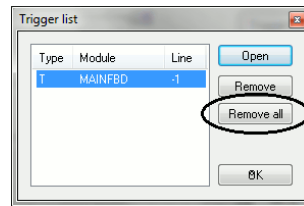
Se avete finito di osservare una serie di variabili tramite una finestra di trigger, si consiglia di chiudere la finestra *Debug* senza rimuovere il trigger.

Se si preme il bottone sull'angolo in altro a destra, si nasconde la finestra, mentre il gestore delle finestre e il relativo trigger continuano a funzionare.

Difatti, se si desidera ripristinare il debug con una finestra trigger precedentemente nascosta, basta aprire la finestra *Trigger list*, selezionare la riga che si riferisce alla finestra trigger e cliccare sul bottone *Open*.

La finestra apparirà con i valori delle variabili e dei contatori dei trigger aggiornati, come se non fosse stata chiusa.

Rimuovere un trigger

Se sceglieste questa opzione, rimuoverete completamente il codice sia dalla finestra che dai suoi trigger. A questo proposito, aprire la finestra *Trigger list*, selezionare la riga corrispondente alla finestra trigger che si vuole eliminare, e premere il bottone *Remove*.

In alternativa, potete spostare il cursore sulla linea (se il modulo è in IL o ST) o cliccare il blocco (se il modulo è in FBD o LD) dove è stato inserito il trigger. Ora premere il bottone *Set/Remove trigger* nella barra *Debug*.

Rimuovere tutti i trigger

Potete rimuovere tutti i trigger esistenti in una volta sola, senza selezionare i record, cliccando sul bottone *Remove all*.

9.6 TRIGGER GRAFICI

9.6.1 FINESTRA DEI TRIGGER GRAFICI

La finestra dei trigger grafici permette di selezionare un insieme di variabili, di campionarle in modo sincrono e di avere la loro curva visualizzata in una speciale finestra pop-up.

Il campionamento delle variabili avviene ogni volta che il processore raggiunge la posizione (cioè le istruzioni - se il linguaggio è IL o ST - o il blocco se il linguaggio è FBD o LD) del trigger.

9.6.1.1 PRE-CONDIZIONI PER APRIRE UNA FINESTRA DI TRIGGER GRAFICI

Nessuna necessità di compilazione dedicata

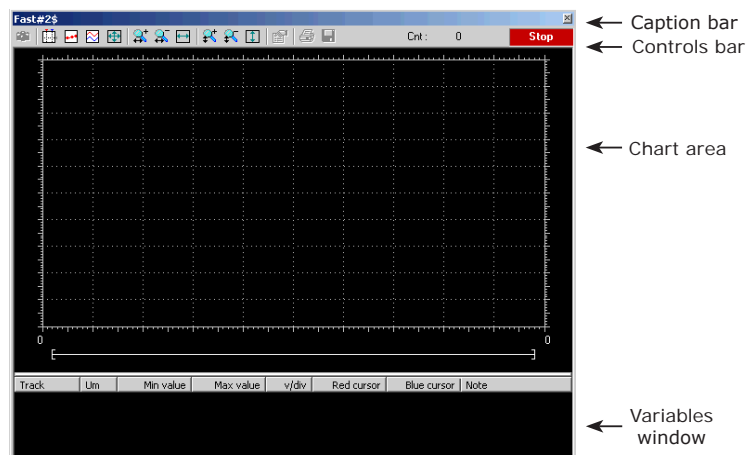
Tutti gli strumenti di debug di LogicLab operano a run-time. Così a differenza di altri linguaggi di programmazione come C++, il compilatore non ha bisogno di sapere se supportare o meno la finestra dei trigger: dato un codice PLC, l'uscita del compilatore è unica, e non vi è alcuna distinzione tra la versione di debug e di rilascio.

Memoria disponibile

Una finestra di trigger grafici richiede tutto lo spazio di memoria libera nel settore del codice dell'applicazione.

Ovviamente per avviare una finestra trigger è necessario che una sufficiente quantità di memoria sia disponibile, altrimenti genererà un errore.

9.6.1.2 INTERFACCIA DELLA FINESTRA DEI TRIGGER GRAFICI

L'impostazione di un trigger grafico provoca l'apertura di una finestra pop-up, chiamata *Interface*. Questa è la stessa interfaccia per accedere alle funzioni di debug che la finestra dei trigger grafici rende disponibile. E' composta da una serie di elementi, come mostrato qui sotto.

The caption bar

La barra *Caption* in cima alla finestra pop-up mostra informazioni sulla posizione del trigger nella quale vengono campionate le variabili elencate nella finestra *Variables*.

Il testo nel titolo ha il seguente formato:

```
ModuleName#Location
```

Dove

`ModuleName`	Nome del programma, funzione, o blocco funzione dove il trigger è stato posizionato.

`Location`	Posizione esatta del trigger, all'interno del modulo `ModuleName`.
	Se `ModuleName` è in IL, ST, `Location` ha il seguente formato:
	`N1`
	Altrimenti, se `ModuleName` è in FBD, LD, diventa:
	`N2$BT:BID`
	`N1` = istruzioni sul numero della riga
	`N2` = numero del network
	`BT` = tipo di blocco (operatore, funzione, blocco funzione, etc.)
	`BID` = identificatore del blocco

La barra dei controlli

Questa finestra di dialogo permette di controllare meglio il lavoro della finestra dei trigger grafici. Una descrizione dettagliata delle funzioni di ogni controllo è data nella sezione Controlli dei trigger grafici (vedi 9.6.1.5).

Area del grafico

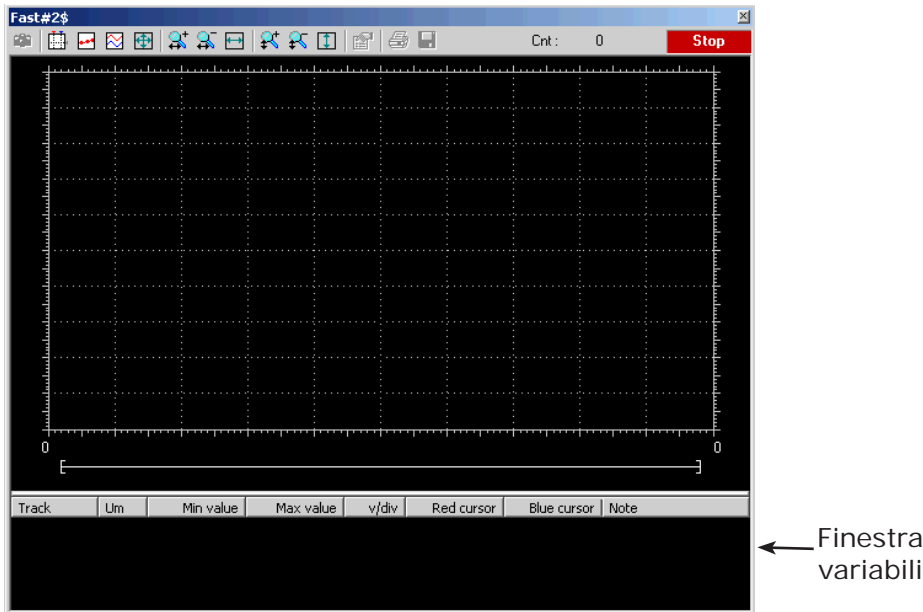
L'area *Chart* include sei oggetti:

1) Plot: area che contiene la traccia attuale della curva delle variabili trascinate.
2) Campioni da acquisire: numero di campioni collezionati con la finestra gestore dei trigger.
3) Cursore orizzontale: il cursore viene identificato con una linea orizzontale. Il valore di ogni variabile all'incrocio con questa linea è riportato nella colonna *horz cursor*.
4) Cursore blu: il cursore viene identificato con una linea verticale. Il valore di ciascuna variabile che si interseca con questa linea è riportato nella colonna *left cursor*.
5) Cursore rosso: il cursore viene identificato con una linea verticale. Il valore di ciascuna variabile che si interseca con questa linea è riportato nella colonna *left cursor*.
6) Barra di scorrimento: se la scala dell'asse x è troppo grande per visualizzare tutti i campioni nell'area *Plot*, la barra di scorrimento consente di scorrere avanti e indietro lungo l'asse orizzontale.

La finestra delle variabili

Questa sezione in basso alla finestra *Debug* è una tabella composta da una riga per ogni variabile. Ogni riga contiene diversi campi descritti nel dettaglio nella sezione di informazione del Drag and drop.

9.6.1.3 FINESTRA TRIGGER GRAFICI: INFORMAZIONI DRAG AND DROP

Per guardare una variabile è necessario copiarla nella sezione inferiore della finestra *Debug*.

Questa sezione inferiore della finestra *Debug* è una tabella composta da una riga per ogni variabile che voi avete trascinato all'interno. Ogni riga ha diversi campi, come mostrato nella figura seguente.

Campo	Descrizione
Track	Nome della variabile.
Um	Unità di misura.
Min value	Valore minimo del set di record.
Max value	Valore massimo del set di record.
Cur value	Valore corrente della variabile.
v/div	How many engineering units are represented by a unit of the y-axis (i.e. the space between two ticks on the vertical axis). Quante unità ingegneristiche sono rappresentate da una unità dell'asse y (cioè lo spazio tra due linee orizzontali del grafico).
Blue cursor	Valore della variabile all'intersezione con la linea identificata con il cursore blu.
Red cursor	Valore della variabile all'intersezione con la linea identificata con il cursore rosso.

Campo	Descrizione
Horz cursor	Valore della variabile all'intersezione con la linea identificata con il cursore orizzontale.

Notare che potete trascinare dentro la finestra dei trigger grafici solo le variabili locali al modulo dove avete posizionato il trigger relativo o le variabili globali o i parametri. Non potete trascinare le variabili dichiarate in un'altro programma o le funzioni o i blocchi funzione.

9.6.1.4 CAMPIONI DI VARIABILI

Considerare il seguente esempio.

Il valore della variabile è campionato tutte le volte che la finestra viene attivata, cioè ogni volta che il processore esegue l'istruzione segnata dalla freccia verde. Tuttavia potete impostare i controlli in modo da avere le variabili campionate anche quando il trigger soddisfa ulteriori condizioni di limiti definiti.

Il valore delle variabili nella colonna *Track* è letta dalla memoria appena prima dell'istruzione segnata e immediatamente dopo l'istruzione precedente.

9.6.1.5 FINESTRA CONTROLLI TRIGGER GRAFICI

Questo paragrafo tratta i controlli della finestra *Graphic trigger*. I controlli permettono di specificare in dettaglio quando LogicLab dovrebbe campionare le variabili aggiunte alla finestra *Variables*.

I controlli della finestra dei trigger grafici agiscono in modo definito sul comportamento della finestra, indipendentemente per il tipo di modulo (IL, ST, FBD o LD) dove il trigger relativo è stato inserito.

L'utente può accedere alla finestra dei controlli attraverso la barra *Controls* della finestra debug.

Bottone	Comando	Descrizione
	Start graphic trace	Quando si preme questo bottone, inizia l'acquisizione. Ora se l'acquisizione è avviata e il bottone viene premuto nuovamente, si arresta il processo di raccolta del campione e si ripristinano i tutti i dati acquisiti fino ad ora.
	Enable/Disable cursors	I due cursori (cursore rosso, cursore blu) possono essere visti e spostati lungo il proprio asse fino a quando il bottone è premuto. Premere nuovamente il bottone se si vogliono nascondere simultaneamente tutti i cursori.
	Show samples	Questo controllo è usato per mettere in evidenza il punto esatto in cui le variabili vengono attivate a ciascun campione.
	Split variables	Quando premuto, questo controllo divide l'asse y in tanti segmenti quante sono le variabili trascinate nell'area, in modo che il diagramma di ciascuna variabile viene disegnato in una banda distinta.

Bottone	Comando	Descrizione
	Show all values	E' usato per riempire la finestra del grafico con tutti i valori campionati per la variabile selezionata nel record corrente.
	Horizontal Zoom In and *Zoom Out*	Lo zoom avanti è un'operazione che rende le curve nell'area *Chart* più grandi sullo schermo, in modo da vedere maggiori dettagli. Lo zoom in dietro è un'operazione che rende le curve più piccole sullo schermo, in modo da poterle vederle per intero. Lo zoom orizzontale agisce solo sull'asse orizzontale.
	Horizontal show all	Questo controllo è usato per centrare orizzontalmente un set di record campionati. Così che il primo campione verrà posizionato sul margine sinistro e l'ultimo sarà posizionato sul margine destro della finestra grafica.
	Vertical Zoom In and *Zoom Out*	*Vertical Zoom* agisce solo sull'asse verticale.
	Vertical show all	Questo controllo è usato per centrare verticalmente un set di record campionati. Così che il valore massimo del campione verrà posizionato vicino al margine in alto e il valore più basso verrà posizionato sul margine inferiore della finestra grafica.
	Graphic trigger window properties	Premendo questo bottone apparirà una finestra di dialogo, che permette di impostare le opzioni generali riguardanti le azioni della finestra dei trigger grafici. Dal momento che le opzioni che potete impostare sono abbastanza numerose, sono trattate in una sezione a parte.
	Print chart	Premere questo bottone per stampare l'area *Chart* e la finestra delle *Variables*.
	Save chart	Premere questo bottone per salvare l'area del grafico.

Contatore trigger

Cnt : 25/100

Questo controllo in sola lettura visualizza due numeri con il seguente formato: X/Y.

X indica quante volte la finestra debug è stata attivata da quando il trigger grafico è stato installato.

Y rappresenta il numero dei campioni che il trigger grafico ha raccolto prima di fermare l'acquisizione dei dati e disegnare le curve.

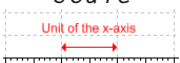
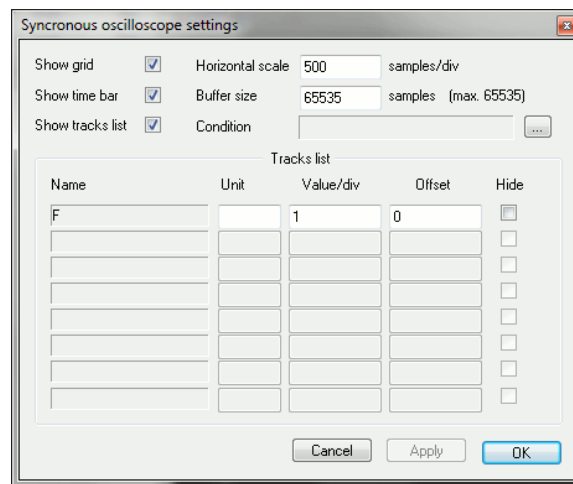
Stato del Trigger

Questo controllo di sola lettura mostra lo stato della finestra *Debug*. Può avere i seguenti valori.

	Nessun campione raccolto, dato che il trigger non si è ancora verificato durante l'esecuzione corrente.

Triggered	Campioni raccolti, il trigger si è verificato durante l'esecuzione dell'attività corrente.
Stop	Il contatore di trigger indica che un numero di campioni è stato raccolto soddisfando la richiesta dell'utente o i vincoli della memoria, quindi il processo di acquisizione viene fermato.
Error	La comunicazione con il target è stata interrotta, lo stato della finestra trigger non può essere determinata.

9.6.1.6 FINESTRA OPZIONI TRIGGER GRAFICO

Per aprire la scheda delle opzioni, premere il bottone *Properties* sulla barra *Controls*. Facendo questo si aprirà la seguente finestra di dialogo.

Generale

Controlli

Controllo	Descrizione
Show grid	Spuntare questo controllo per visualizzare una griglia sullo sfondo dell'area *Chart*.
Show time bar	La barra di scorrimento nella parte inferiore dell'area *Chart* è disponibile fino a quando questo box sarà spuntato.
Show tracks list	La finestra delle *Variables* è mostrata fino a quando questo box sarà spuntato, altrimenti l'area *Chart* si estenderà fino alla fine della finestra del trigger grafico.

Valori

Controllo	Descrizione
Horizontal scale Unit of the x-axis	Numero di campioni per unità dell'asse x. Con unità dell'asse x si intende lo spazio tra due linee verticali sulla griglia di sfondo.

Controllo	Descrizione
Buffer size	Numero di campioni da acquisire. Quando si apre la scheda delle opzioni, dopo avere trascinato tutte le variabili che si vogliono controllare, in questo campo si può leggere un numero di default, che rappresenta il numero massimo dei campioni che si possono raccogliere per ogni variabile. E' quindi possibile digitare un numero maggiore o inferiore a quello di default.

Tracce

Questa scheda permette di definire alcune proprietà grafiche della traccia di ciascuna variabile. Per selezionare una variabile, cliccare sul nome nella colonna *Track list*.

Controllo	Descrizione
Unit	Unità di misura, stampata sulla tabella della finestra delle *Variables*.
Value/div	Δ valore per unità dell'asse y. Con unità dell'asse y si intende lo spazio tra due linee orizzontali sulla griglia di sfondo.
Hide	Selezionare questa opzione per nascondere le tracce selezionate sul grafico.

Premere *Apply* per rendere le modifiche effettive, o premere *OK* per applicare le modifiche e chiudere la finestra.

Condizioni definite dall'utente

Se si definisce una condizione usando questo controllo, il processo di campionamento non partirà fino a quando questa condizione non sarà soddisfatta. Notare che, a differenza della finestra trigger, una volta iniziata l'acquisizione di dati, i campioni sono prelevati ogni volta che viene attivata la finestra, indipendentemente che la condizione sia vara o no.

Dopo avere inserito una condizione, il controllo visualizza la sua espressione semplificata.

9.6.2 DEBUG CON LA FINESTRA DEI TRIGGER GRAFICI

Lo strumento della finestra di trigger grafici permette di selezionare un insieme di variabili, di averle campionate in modo sincrono e di avere le loro curve visualizzate in una speciale finestra.

9.6.2.1 APRIRE UNA FINESTRA DI TRIGGER GRAFICI CON UN MODULO IL

Supponiamo di avere un modulo IL, e che contenga le seguenti istruzioni.

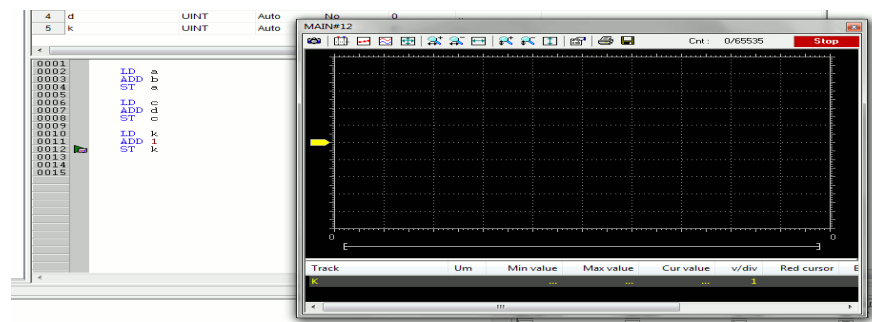
Supponiamo inoltre di volere conoscere il valore di b, d, e k, appena prima che venga eseguita l'istruzione ST *k*. Per fare ciò, spostare il cursore alla linea 12.

Quindi premere il bottone *Graphic trace* sulla barra *Debug.*

Una freccia verde appare vicino al numero della linea le la finestra di trigger grafici appare.

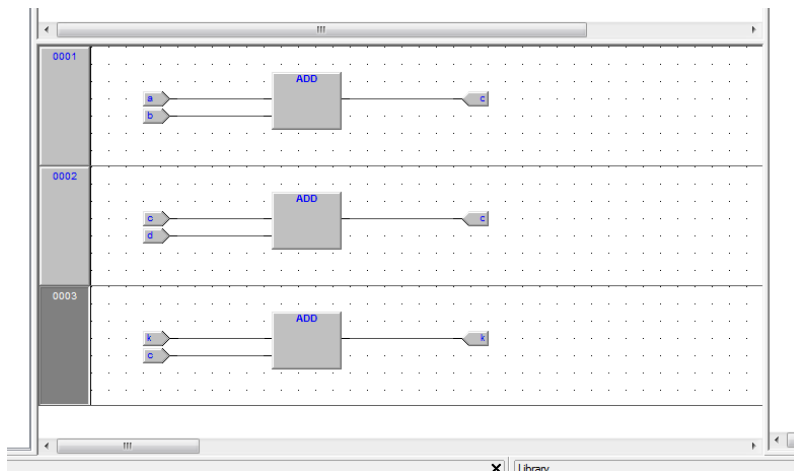
Non tutte le istruzioni IL supportano i trigger. Per esempio, non è possibile posizionare un trigger all'inizio di una linea che contiene un'istruzione JMP.

9.6.2.2 AGGIUNGERE UNA VARIABILE AD UNA FINESTRA DI TRIGGER GRAFICI DA UN MODULO IL

Per ottenere il diagramma di una variabile tracciata, è necessario aggiungerla nella finestra dei trigger grafici. A questo scopo, selezionare una variabile, facendo doppio click su di essa e trascinarla nella finestra *Variables.* La variabile ora apparirà nella colonna *Track.*

Applicare la stessa procedura per tutte le variabili che si vogliono controllare.

Una volta che la prima variabile viene immessa in una traccia grafica, si apre automaticamente la finestra *Graphic properties* e permette all'utente di impostare campioni e visualizzare proprietà.

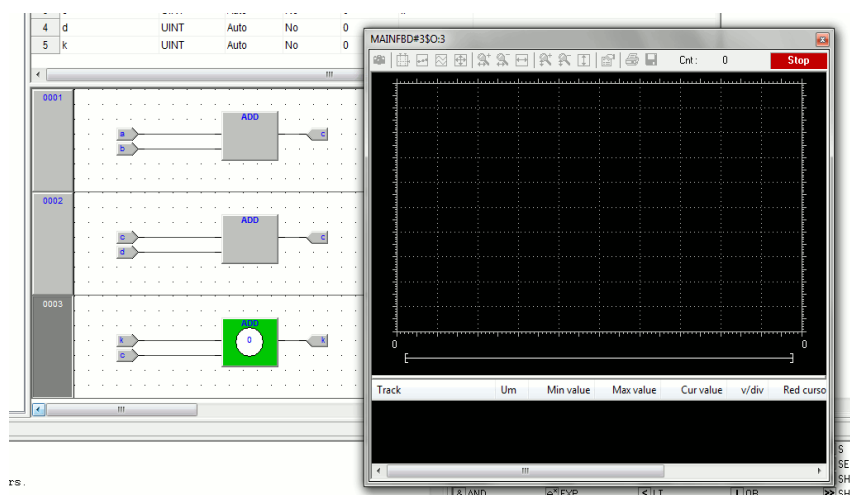
9.6.2.3 APRIRE LA FINESTRA DI TRIGGER GRAFICI DA UN MODULO FBD

Supponiamo di avere un modulo FBD, e che contenga le seguenti istruzioni.

Supponiamo inoltre di voler conoscere il valore di `c`, `d`, e `k`, appena prima che venga eseguita l'istruzione *ST k*.

Non si può mai impostare un trigger in un blocco che rappresenta una variabile come questa

selezionare il primo blocco disponibile precedente alla variabile selezionata. Nell'esempio della figura sopra, si deve spostare il cursore al network 3 e premere il blocco `ADD`.

Ora premere il bottone *Graphic trace* nella barra *Debug*.

Questo provoca il cambiamento di colore del blocco selezionato che diventa verde, un cerchio verde con il numero ID del trigger appare nel mezzo del blocco e la relativa finestra trigger appare.

Quando si preprocessa il codice sorgente FBD, il compilatore traduce le istruzioni in linguaggio IL. L'istruzione *ADD* al network 3 è ampliata a:

```
LD  k
ADD 1
ST  k
```

Quando si aggiunge un trigger in un blocco FBD, si posiziona effettivamente il trigger prima della prima istruzione del suo codice equivalente IL.

9.6.2.4 AGGIUNGERE UNA VARIABILE AD UNA FINESTRA DI TRIGGER GRAFICI DA UN MODULO FBD

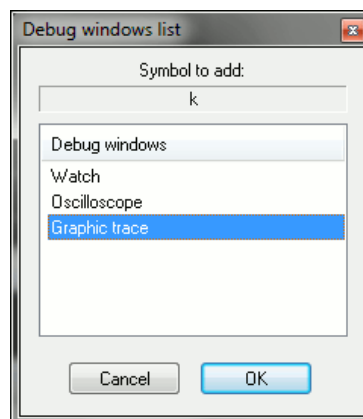
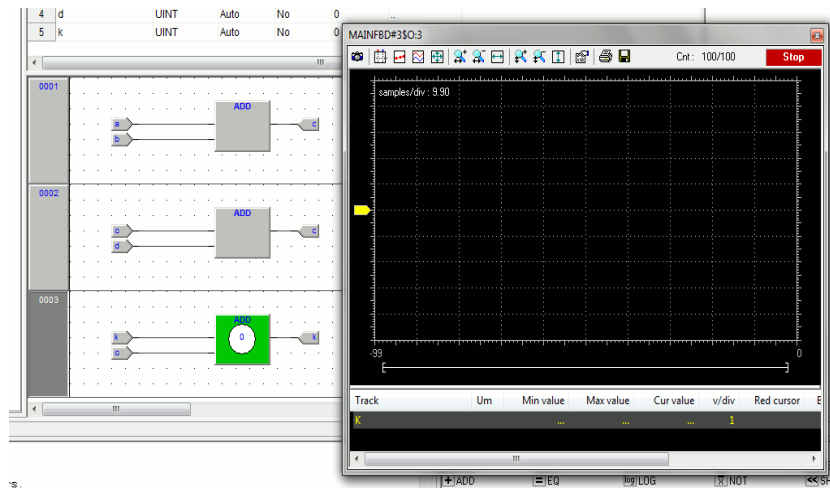
Per guardare il diagramma di una variabile, è necessario aggiungerla alla finestra trigger. Supponiamo di voler vedere la traccia della variabile *k* del codice FBD nella figura seguente.

A questo proposito, premere il bottone *Watch* nella barra FBD.

Il cursore diventa come il seguente.

Ora si può premere il blocco che rappresenta la variabile che si vuole vedere nella finestra dei trigger grafici.

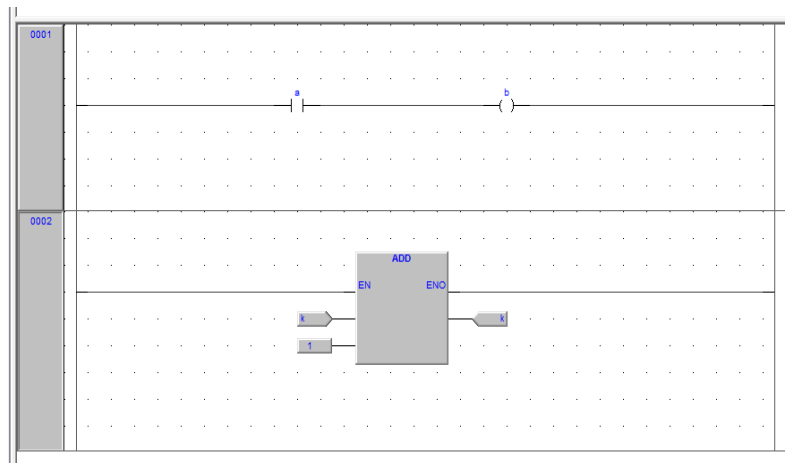
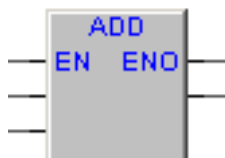
Nell'esempio che stiamo considerando fare click sul blocco.

Appare una finestra di dialogo che elenca tutte le istanze attualmente esistenti della finestra debug e chiede quale di queste deve ricevere l'oggetto che avete appena cliccato.

Per tracciare le curve della variabile *k*, selezionare *Graphic Trace* nella colonna *Debug windows*, quindi premere *OK*. Il nome della variabile è ora stampato nella colonna *Track*.

Applicare la stessa procedura a tutte le variabili che si vogliono controllare.

Una volta aggiunte alla finestra `Graphic watch` tutte le variabili che si vogliono osservare, potete premere il bottone `Normal cursor`, per tornare al cursore originale.

Una volta che la prima variabile viene immessa in una traccia grafica, la finestra `Graphic properties` è automaticamente mostrata e permette all'utente di impostare i campioni e visualizzare proprietà.

9.6.2.5 APRIRE UNA FINESTRA DI TRIGGER GRAFICI DA UN MODULO LD

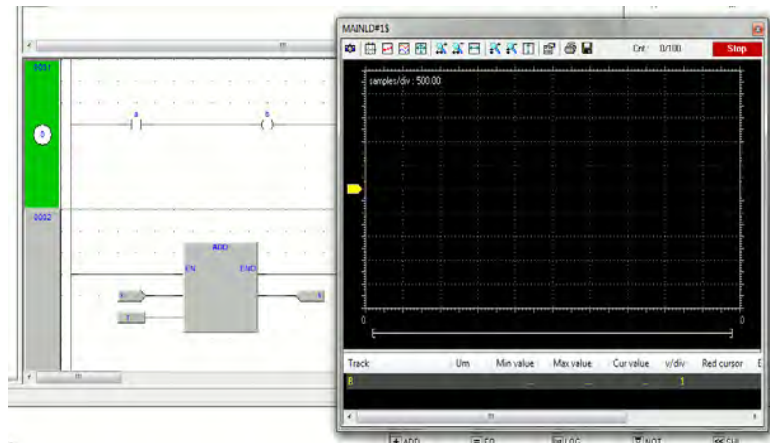
Supponiamo di avere un modulo LD e che contenga le seguenti istruzioni.

Potete posizionare un trigger in un blocco come il seguente.

In questo caso applicare le stesse regole usate per inserire un trigger grafico in un modulo FBD su un contatto

o uscita

-( )-

In questo caso, seguire le istruzioni. Supponiamo di voler conoscere il valore di alcune variabili tutte le volte che il processore raggiunge il network numero 1.

Fare click su undo degli elementi che compongono il nerwork nr. 1, quindi premere il bottone *Graphic trace* sulla barra *Debug.*

Questo provoca il cambio di colore del bottone grigio in rilievo contenente il numero del network, che diventa verde, un cerchio bianco con all'interno un numero appare nel mezzo del bottone e la finestra dei trigger grafici appare.

Notare che a differenza degli altri linguaggi supportati da LogicLab, LD non permette di inserire un trigger prima di un singolo contatto o un uscita, in quanto permette di selezionare solo un network intero. Così le variabili nella finestra *Graphic trigger* sarà campionata tutte le volte che il processore raggiungerà l'inizio del network selezionato.

9.6.2.6 AGGIUNGERE UNA VARIABILE AD UNA FINESTRA DI TRIGGER GRAFICI DA UN MODULO LD

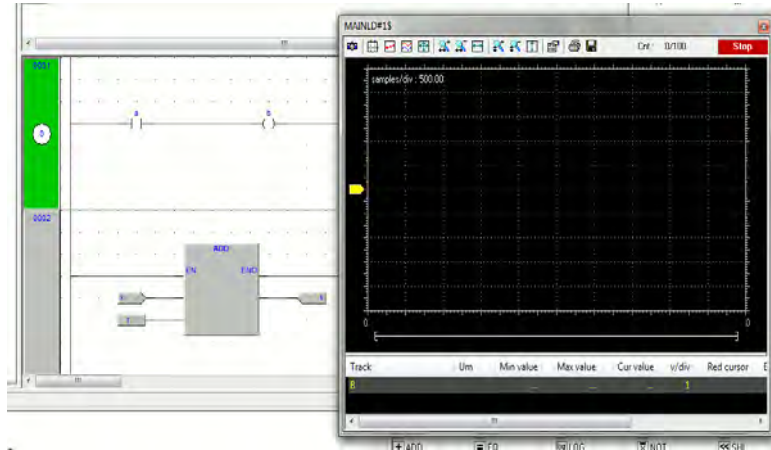
Per vedere il diagramma di una variabile, è necessario aggiungerla alla finestra *Graphic trigger.* Supponiamo che si vuole vedere la traccia della variabile b nel codice LD rappresentato nella figura sotto.

A questo proposito, premere il bottone *Watch* nella barra FBD.

Il cursore diventerà come il seguente.

Ora potete premere l'oggetto che rappresenta la variabile che si vuole mostrare nella finestra *Graphic trigger*.

Apparirà una finestra di dialogo con l'elenco di tutte le istanze esistenti della finestra debug e chiede quale di queste deve ricevere l'oggetto che avete appena cliccato.

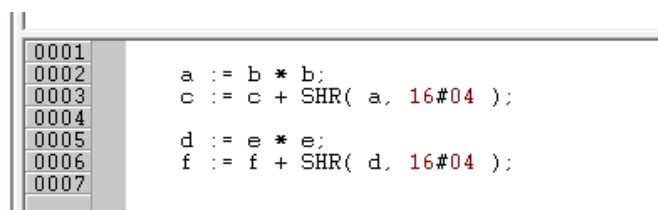
Per tracciare le curve della variabile *b*, selezionare *Graphic trace* nella colonna *Debug windows*, quindi premere *OK*. Il nome della colonna è ora stampato sulla colonna *Track*.

Applicare la stessa procedura a tutte le variabili che si vogliono controllare.

Una volta aggiunte tutte le variabili che si vogliono osservare alla finestra *Graphic watch*, si può premere nuovamente sul bottone *Normal cursor*, in modo da ripristinare la forma originale del cursore.

Una volta che la prima variabile è stata trascinata in una traccia grafica, la finestra *Graphic properties* è automaticamente mostrata e permette di impostare i campioni e visualizzare proprietà.

9.6.2.7 APRIRE UNA FINESTRA DI TRIGGER GRAFICI DA UN MODULO ST

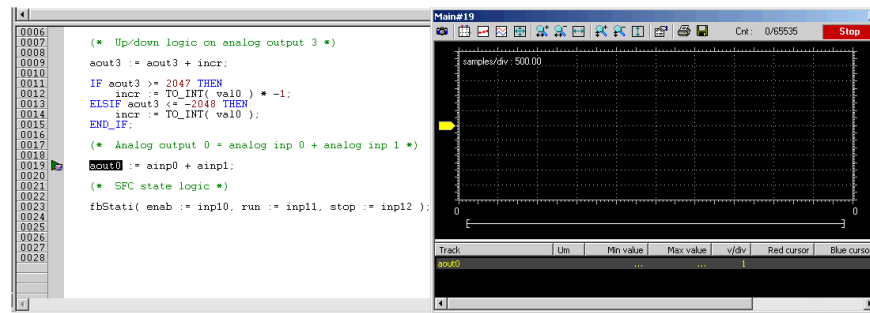
Supponiamo di avere un modulo ST e che contenga le seguenti istruzioni.

```
0001
0002    a := b * b;
0003    c := c + SHR( a, 16#04 );
0004
0005    d := e * e;
0006    f := f + SHR( d, 16#04 );
0007
```

Supponiamo inoltre di voler conoscere il valore di `e`, `d`, e `f`, appena prima che l'istruzione

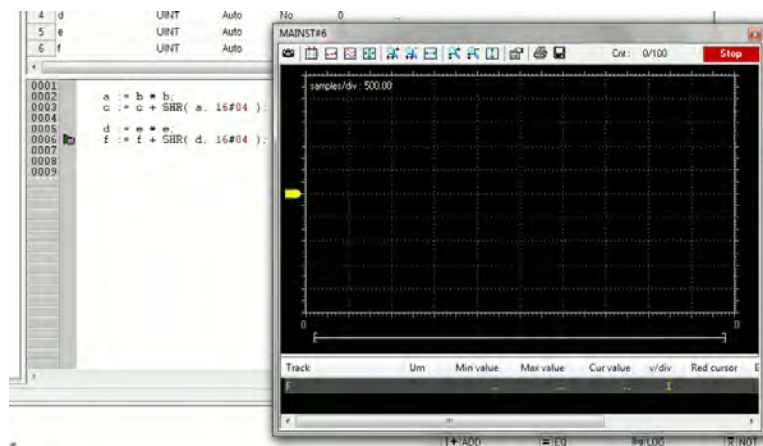
```
f := f+ SHR( d, 16#04 )
```

sia eseguita. Per fare questo,spostare il cursore alla linea 6.

Quindi premere il bottone *Graphic trace* sulla barra *Debug*.

Una freccia verde appare vicino al numero della linea, e la finestra *Graphic trigger* si apre.

Non tutte le istruzioni St supportano i trigger. Per esempio, non è possibile posizionare un trigger su una linea che contiene un terminatore come END_IF, END_FOR, END_WHILE, etc.

9.6.2.8 AGGIUNGERE UNA VARIABILE AD UNA FINESTRA DI TRIGGER GRAFICI DA UN MODULO ST

Per ottenere il diagramma di una variabile tracciata, è necessario aggiungerla alla finestra *Graphic trigger.* A questo proposito, selezionare la variabile, facendo doppio click e quindi trascinarla nella finestra *Variables,* che è la parte bianca in basso della finestra pop-up. La variabile ora appare nella colonna Track.

Applicare la stessa procedura a tutte le variabili che si vogliono controllare.

Una volta che la prima variabile è stata trascinata in una traccia grafica, la finestra *Graphic properties* è automaticamente mostrata e permette di impostare i campioni e visualizzare proprietà.

9.6.2.9 RIMUOVERE UNA VARIABILE DA UNA FINESTRA DI TRIGGER GRAFICI

Se si vuole rimuovere una variabile dalla finestra *Graphic trigger*, selezionarla cliccando sul suo nome, quindi premere il tasto *Del*.

9.6.2.10 USARE I CONTROLLI

Questo paragrafo tratta i controlli della finestra dei trigger grafici, che permette di migliorare il controllo del lavoro dello strumento debug, in modo da ottenere ulteriori informazioni sul codice sotto controllo.

Abilitare i controlli

Quando si imposta un trigger, tutti gli elementi sulla barra *Control* sono abilitati. Potete iniziare l'acquisizione dei dati cliccando il bottone *Start graphic trace acquisition.*

Se si definisce una condizione, che è attualmente falsa, l'acquisizione dei dati non parte, anche se si preme l'apposito pulsante.

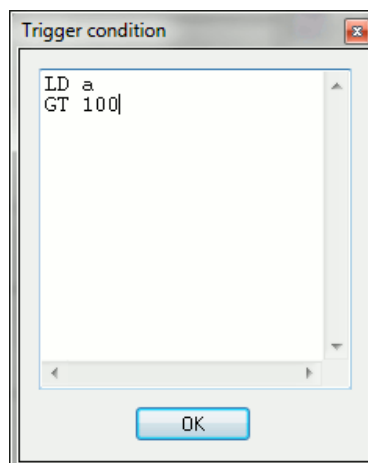
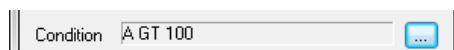
Al contrario, una volta che la condizione diventa vera, l'acquisizione inizia e continua fino a quando non viene rilasciato il bottone *Start graphic trace acquisition*, indipendentemente dalla condizione che sia ancora vera o no.

Se si rilascia il bottone *Start graphic trace acquisition* prima che tutti i campioni necessari siano stati acquisiti, il processo di acquisizione si ferma e tutti i dati raccolti saranno persi.

Definire una condizione

Questo controllo consente all'utente di impostare una condizione su quando iniziare l'acquisizione. Per impostazione predefinita, questa condizione è impostata a true e l'acquisizione inizia non appena si preme il bottone *Enable/Disable acquisition*. Da questo momento il valore delle variabili nella finestra *Debug* è campionato tutte le volte che si verifica il trigger.

Per specificare una condizione, aprire la scheda *Condition* della finestra di dialogo *Options*, quindi premere il seguente bottone.

Si apre una finestra di testo, dove poter scrivere il codice IL che imposta le condizioni.

Una volta finito di scrivere il codice della condizione, premere il bottone *OK* per installare, o premere il bottone *Esc* per cancellare. Il campionamento non partirà fino a quando il bottone *Start graphic trace acquisition* sarà premuto e le condizioni definite dall'utente saranno vere. Un' espressione semplificata della condizione appare ora nel controllo.

Per modificarla premere nuovamente il relativo bottone.

La finestra di testo appare, contenente il testo originariamente scritto, che potete ora modificare. Per rimuovere completamente una condizione definita dall'utente, premere nuovamente sul pulsante menzionato, eliminare l'intero codice IL nella finestra di testo, quindi premere *OK*.

Dopo l'esecuzione della condizione, l'accumulatore deve essere di tipo Boolean (*TRUE* o *FALSE*), in caso contrario si verifica un errore del compilatore. Possono essere usate nel codice della condizione solo variabili globali e le variabili trascinate della finestra.

Vale a dire che tutte le variabili locali al modulo dove il trigger è stato originariamente inserito sono fuori portata se non sono state precedentemente trascinate nella finestra *Debug.* Inoltre non possono essere dichiarate nuove variabili.

Impostare la scala degli assi

- x-axis

Quando l'acquisizione è completata, LogicLab traccia le curve delle variabili trascinate regolando l'asse x in modo che tutti i dati siano inseriti nella finestra *Chart.* Se si vuole applicare una scala differente, aprire la scheda *General* della finestra di dialogo *Graph properties*, digitare un numero nella cella relativa la scala orizzontale, quindi confermare premendo *Apply.*

- y-axis

E' possibile modificare la scala della traccia di ciascuna variabile attraverso la scheda *Tracks list* della finestra di dialogo *Graph properties.* Altrimenti, se non si necessita di specificare esattamente una scala, potete usare i controlli *Zoom In* e *Zoom Out.*

9.6.2.11 CHIUDERE LA FINESTRA DEI TRIGGER GRAFICI E RIMUOVERE IL TRIGGER

Alla fine di una sessione di debug con la finestra dei trigger grafici potete scegliere tra le seguenti opzioni:

- Chiudere la finestra *Graphic trigger*.
- Rimuovere il trigger.
- Rimuovere tutti i trigger.

Chiudere la finestra dei trigger grafici

Se avete finito di tracciare il diagramma di un insieme di variabili per mezzo della finestra *Graphic trigger*, si consiglia di chiudere la finestra di *Debug* senza rimuovere il trigger. Se si preme il bottone nell'angolo in alto a destra, si nasconde la finestra *Interface*, mentre la finestra manager e il relativo trigger continuano a lavorare.

Difatti, se poi si vuole ripristinare la finestra *Graphic trigger* che si nascondeva in precedenza:

- aprire la finestra *Trigger list*;
- selezionare il record (tipo *G*);
- cliccare il bottone *Open*.

La finestra *Interface* appare con il contatore del trigger correttamente aggiornato, come se non fosse mai stato chiuso.

Rimuovere un trigger

Se si sceglie questa opzione, si rimuove completamente il codice sia della finestra manager che del suo trigger. A questo proposito:

- aprire la finestra *Trigger list*;
- selezionare il record (tipo G);
- premere il bottone *Remove*.

Oppure, si può spostare il cursore sulla linea (se il modulo è in IL), o cliccare sul blocco (se il codice è in FBD) dove avete posizionato il trigger. Ora premere il bottone *Graphic trace* nella barra *Debug.*

Rimuovere tutti i trigger

In alternativa, potete rimuovere tutti i trigger in una sola volta, indipendentemente dai record selezionati, cliccando sul bottone *Remove all triggers.*

10. ELEMENTI LOGICLAB

10.1 ELEMENTI DEI MENU

Nelle tabelle seguenti è riportata la lista di tutti i comandi LogicLab. Tuttavia, poiché LogicLab ha un'interfaccia multi-document (MDI), si possono trovare alcuni comandi disabilitati o addirittura menu non disponibili, a seconda di quale documento è correntemente attivo.

10.1.1 MENU FILE

Comando	Descrizione
New project	Permette di creare un nuovo progetto LogicLab.
Open project	Permette di aprire un progetto LogicLab esistente.
View project	Apre un progetto LogicLab esistente in sola lettura.
Save project	Uguale a *Save All*, ma salva anche il file *ppj*. Notare che tutte le modifiche al progetto LogicLab sono prima applicate solo in memoria, è necessario eseguire il comando *Save Project* per rendere le modifiche permanenti.
Save project As	Richiede di specificare il nuovo nome e la nuova posizione di un progetto, e salva qui una copia di tutti i file del progetto.
Close project	Richiede se si vogliono salvare i cambiamenti, poi chiude il progetto attivo.
New text file	Apre un nuovo file generico di testo vuoto.
Open file	Apre un file esistente, qualunque sia la sua estensione. Il file è mostrato nell'editor testuale. Comunque, aprendo un file del progetto, si apre anche il progetto LogicLab a cui questo è riferito.
Save	Permette di salvare il documento della finestra correntemente attiva.
Close	Chiude il documento della finestra correntemente attiva.
Options	Apre la finestra di dialogo *Programming environment options*.
Print	Mostra una finestra di dialogo che permette di impostare le opzioni di stampa e stampare il documento della finestra correntemente attiva.
Print preview	Mostra un'immagine nel video che riproduce fedelmente quello che si ottiene stampando il documento della finestra correntemente attiva.
Print project	Stampa tutti i documenti che compongono il progetto.
Printer setup	Apre la finestra di dialogo *Printer setup*.
..recent..	Elenca una serie di file ppj di progetti LogicLab aperti recentemente. Cliccare su uno di essi per aprire il progetto corrispondente.
Exit	Chiude LogicLab.

10.1.2 MENU EDIT

Comando	Descrizione
Undo	Cancella l'ultima modifica apportata al documento.
Redo	Ripristina l'ultima modifica annullata da *Undo*.
Cut	Rimuove le voci selezionate dal documento attivo e le salva in un buffer di sistema.
Copy	Copia le voci selezionate ad un buffer di sistema
Paste	Incolla nel documento attivo i contenuti del buffer di sistema.
Delete	Elimina gli oggetti selezionati.
Delete line	Cancella l'intera linea di codice sorgente
Find in project	Apre la finestra di dialogo *Find in project*.
Bookmarks	Permette di impostare, rimuovere e spostarsi tra i segnalibri.
Go to line	Permette di spostarsi velocemente ad una linea specifica dell'editor di codice sorgente.
Find	Chiede di digitare una stringa e ricerca le sue ricorrenze all'interno del documento attivo dalla posizione corrente del cursore.
Find next	Continua la ricerca precedentemente compiuta dal comando Find.
Replace	Permette di sostituire automaticamente una o tutte le ricorrenze di una stringa con un'altra.
Insert/Move mode	Modalità di modifica che permette di inserire e spostare i blocchi.
Connection mode	Modalità di modifica che permette di disegnare fili logici per connettere i pins.
Watch mode	Modalità di modifica che permette di aggiungere variabili a qualsiasi strumento di debug.

10.1.3 MENU VIEW

Comando	Descrizione
Main Toolbar	Se selezionato mostra la barra *Main*, altrimenti la nasconde.
Status bar	Se selezionato mostra la barra *Status*, altrimenti la nasconde.
Debug bar	Se selezionato mostra la barra *Debug*, altrimenti la nasconde.
FBD bar	Se selezionato mostra la barra *FBD*, altrimenti la nasconde.
LD bar	Se selezionato mostra la barra *LD*, altrimenti la nasconde.
SFC bar	Se selezionato mostra la barra *SFC*, altrimenti la nasconde.
Project bar	Se selezionato mostra la barra *Project*, altrimenti la nasconde.
Network	Se selezionato mostra la barra *Network*, altrimenti la nasconde.
Document bar	Se selezionato mostra la barra *Document*, altrimenti la nasconde.

Comando	Descrizione
Force I/O bar	Se selezionato mostra la barra *Force I/O*, altrimenti la nasconde.
Workspace	Se selezionato mostra la finestra *Workspace* (chiamata anche finestra *Project*), altrimenti la nasconde.
Library	Se selezionato mostra la finestra *Libraries*, altrimenti la nasconde.
Output	Se selezionato mostra la finestra *Output*, altrimenti la nasconde.
Async Graphic window	Se selezionato mostra la finestra *Oscilloscope*, altrimenti la nasconde.
Watch window	Se selezionato mostra la finestra *Watch*, altrimenti la nasconde.
Full screen	Espande la finestra del documento attivo a tutto schermo. Premere *Esc* per tornare alla normale interfaccia di LogicLab.
Grid	Se selezionato mostra una griglia sullo sfondo di codice sorgente grafico.

10.1.4 MENU PROJECT

Comando	Descrizione
New object	Apre un altro menu che permette di creare una nuova POU o dichiarare una nuova variabile globale.
Copy object	Copia l'oggetto correntemente selezionato nel Workspace.
Paste object	Incolla l'oggetto precedentemente copiato.
Duplicate object	Duplica l'oggetto correntemente selezionato nel Workspace, e chiede di digitare il nome della copia.
Delete object	Cancella l'oggetto correntemente selezionato. Come spiegato sopra, è necessario eseguire il comando *Save project* per eliminare definitivamente un documento dal progetto.
PLC object properties	Mostra le proprietà e la descrizione dell'oggetto correntemente selezionato nel *Workspace*.
Object browser	Apre l'*Object browser*, che permette di navigare tra gli oggetti.
Compile	Chiede se si vogliono salvare le modifiche, poi lancia il compilatore di LogicLab.
Recompile all	Ricompila il progetto.
Generate redistributable source module	Genera un file RSM.
Import object from library	Permette di importare un oggetto LogicLab da una libreria.
Export object to library	Permette di esportare un oggetto LogicLab ad una libreria.
Library manager	Apre *Library manager*.
Macros	Apre un altro menu che permette di creare/cancellare macro.
Select target	Permette di cambiare il target.

Comando	Descrizione
Options...	Permette di specificare le opzioni del progetto.

10.1.5 MENU DEBUG

Comando	Descrizione
Add symbol to watch	Aggiunge un simbolo alla finestra *Watch*.
Insert new item into watch	Inserisce un nuovo elemento nella finestra *Watch*.
Add symbol to a debug window	Aggiunge un simbolo alla finestra debug.
Insert new item into a debug window	Inserisce un nuovo elemento in una finestra di debug.
Quick watch	Apre una finestra con il valore attuale della variabile.
Run	Fa ripartire il programma dopo un breakpoint.
Add/Remove breakpoint	Aggiunge o rimuove un breakpoint.
Remove all breakpoints	Rimuove tutti i breakpoint attivi.
Breakpoint list	Elenca tutti i breakpoint attivi.
Add/remove text trigger	Aggiunge rimuove un trigger testuale.
Add/remove graphic trigger	Aggiunge rimuove un trigger grafico.
Remove all triggers	Rimuove tutti i trigger attivi.
Trigger list	Elenca tutti i trigger attivi.
Debug mode	Attiva la modalità debug.
Live debug mode	Attiva la modalità live debug.

10.1.6 MENU COMMUNICATION

Comando	Descrizione
Download code	LogicLab controlla se sono stati applicati dei cambiamenti dall'ultima compilazione, e in tal caso compila il progetto. Poi, invia il codice compilato al target.
Connect	LogicLab cerca di stabilire una connessione con il target.
Settings	Permette di impostare le proprietà di connessione al target.
Upload IMG file	Se il target device è connesso, permette di caricare il file *img*.
Start/Stop watch value	Blocca/sblocca l'aggiornamento della finestra *Watch*.

10.1.7 MENU SCHEME

Comando	Descrizione
Network> New> Top	Aggiunge un network vuoto nella parte alta del documento LD/FBD attivo.
Network> New> Bottom	Aggiunge un network vuoto nella parte bassa del documento LD/FBD attivo.
Network> New> Before	Aggiunge un network vuoto prima del network selezionato nel 138 documento LD/FBD attivo.
Network >New > After	Aggiunge un network vuoto dopo il network selezionato nel documento LD/FBD attivo.
Network >Label	Assegna un'etichetta ad un network selezionato, per essere indicato come target di un'istruzione jump.
Object >New	Permette di inserire un nuovo oggetto in un network selezionato.
Object > Instance name	Permette di assegnare un nome ad una ricorrenza di un blocco funzione, precedentemente selezionato con un click.
Object > Open source	Apre l'editor con cui l'oggetto è stato creato e mostra il codice sorgente corrispondente: - Se l'oggetto è un programma, una funzione o un blocco funzione, questo comando apre il suo codice sorgente; - se l'oggetto è una variabile o un parametro, questo comando apre l'editor di variabili corrispondente; - se l'oggetto è una funzione standard o un operatore, questo comando non apre niente.
Auto connect	Se selezionato, abilita l'autoconnessione, che è una creazione automatica di un filo logico che collega i pins di due blocchi, quando sono sistemati vicini.
Delete invalid connection	Rimuove tutte le connessioni non valide, rappresentate da una linea rossa nello schema attivo.
Increment pins	Per default, alcuni operatori come ADD, MUL, ecc..hanno due pins di input, anche se può esserci occasionalmente bisogno di compiere queste operazioni su più di due operandi. Questo comando permette di aggiungere tanti pins di input quanti ne sono richiesti per raggiungere il numero voluto di operandi.
Decrement pins	Annulla il comando *Increment* pins.
Enable EN/ENO pins	Aggiunge i pins *enable in*/*enable out* al blocco selezionato. Il codice che implementa il blocco selezionato sarà eseguito solo quando il segnale *enable in* è true. Il segnale *enable out* ripete semplicemente il valore di *enable in*, permettendo sia di abilitare sia di disabilitare il set di blocchi in sequenza.
Object properties	Mostra alcune proprietà del blocco selezionato: - Se l'oggetto è una funzione o un blocco funzione, mostra una tabella con le variabili input e output; - Se l'oggetto è una variabile o un parametro, apre una finestra di dialogo che permette di cambiarne il nome e la direzione logica (input/output).

10.1.8 MENU VARIABLES

Comando	Descrizione
Insert	Aggiunge una nuova riga alla tabella nell'editor correntemente attivo (se è un editor PLC, alla tabella delle variabili locali;se è un editor di variabili, alla tabella dei parametri,ecc.).
Delete	Cancella la variabile nella riga selezionata della tabella correntemente attiva.
Group	Apre una finestra di dialogo che permette di creare e cancellare gruppi di variabili.

10.1.9 MENU DEFINITIONS

Comando	Descrizione
Insert> Enum	Crea un nuovo tipo di dato enumerativo.
Insert> Structure	Crea un nuovo tipo di dato strutturato.
Insert> Subrange	Crea un nuovo tipo di dato subrange.
Insert> Typedef	Crea un nuovo tipo di dato typedef.

10.1.10 MENU WINDOW

Comando	Descrizione
Cascade	Sposta tutti i documenti aperti in sequenza, così che si sovrappongano completamente eccetto che per la loro didascalia.
Tile	L'area dell'editor PLC è divisa in riquadri aventi le stesse dimensioni, che dipendono dal numero di documenti correntemente aperti. Ogni riquadro è automaticamente assegnato ad uno di questi documenti.
Arrange Icons	Sposta le icone dei documenti ridotti a icona nell'angolo in basso a sinistra dell'area dell'editor PLC.
Close all	Chiude tutti i documenti aperti.

10.1.11 MENU HELP

Comando	Descrizione
Index	Elenca tutte le *Help keyword* e apre l'argomento relativo.
Context	Aiuto relativo al contesto. Apre l'argomento corrispondente alla finestra correntemente attiva.
About...	Informazione sul produttore e la versione.

10.2 ELEMENTI TOOLBARS

Le tabelle seguenti mostrano l'elenco delle barre d'applicazione di LogicLab. I tasti componenti ciascuna barra sono sempre gli stessi, qualunque sia il documento attivo. Tuttavia,

alcuni di questi potrebbero essere nulli, se non c'è una relazione logica col documento attivo.

10.2.1 TOOLBAR PRINCIPALE

Bottone	Comando	Descrizione
	New project	Crea un nuovo progetto.
	Open project	Apre un progetto esistente.
	Save project	Salva tutti i documenti della finestra correntemente aperta, compreso il file progetto. Notare che, dal momento che tutte le modifiche ad un progetto LogicLab sono dapprima applicate solo in memoria, è necessario eseguire il comando *Save project* per renderli permanenti.
	Undo	Cancella l'ultima modifica apportata al documento.
	Redo	Ripristina l'ultima modifica cancellata da *Undo*.
	Cut	Rimuove le voci selezionate dal documento attivo e le mette in un buffer di sistema.
	Copy	Copia le voci selezionate in un buffer di sistema.
	Paste	Incolla nel documento attivo i contenuti del buffer di sistema.
	Find	Chiede di digitare una stringa e cerca le ricorrenze di quest'ultima all'interno del documento attivo dalla posizione in cui si trova il cursore.
	Find next	Prosegue la ricerca precedentemente compiuta dal comando *Find*.
	Find in project	Apre la finestra di dialogo *Find in project*.
	Print	Mostra una finestra di dialogo, che permette di impostare le opzioni di stampa e stampare il documento correntemente aperto.
	Print preview	Mostra un'immagine sul video, che riproduce fedelmente cosa si otterrebbe stampando il documento della finestra correntemente attiva.
	Workspace	Se premuto, mostra il *Workspace* (anche chiamato finestra *Project*), altrimenti la nasconde..
	Output	Se premuto, visualizza la finestra di *Output*, altrimenti la nasconde.
	Library	Se premuto, mostra la finestra *Libraries*, altrimenti la nasconde.
	Watch	Se selezionato, mostra la finestra *Watch*, altrimenti la nasconde.

Bottone	Comando	Descrizione
	Async	Se selezionato, mostra la finestra *Oscilloscope*, altrimenti la nasconde
	Force I/O	Se premuto, mostra la finestra *Force I/O*, altrimenti la nasconde.
	PLC run-time monitor	Se selezionato, visualizza la finestra PLC run-time, altrimenti la nasconde.
	Full screen	Espande la finestra del documento correntemente attivo a schermo intero. Premere Esc o il tasto Full screen per tornare all'aspetto normale dell'interfaccia LogicLab.

10.2.2 TOOLBAR FBD

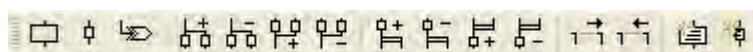
Bottone	Comando	Descrizione
	Move/Insert	Modalità edit che permette di inserire e spostare blocchi.
	Connection	Modalità edit che permette di tracciare fili logici per connettere i pins.
	Watch	Modalità edit che permette di aggiungere variabili a qualsiasi strumento di debug.
	New block	Permette di inserire un nuovo blocco nel network selezionato
	Constant	Aggiunge una costante al network selezionato.
	Return	Aggiunge un blocco di ritorno condizionale al network selezionato.
	Jump	Aggiunge un blocco di jump condizionale al network selezionato.
	Comment	Aggiunge un commento al network selezionato.
	Inc pins	Per default alcuni operatori come *ADD, MUL*, ecc. Hanno due pin di input, anche se si può occasionalmente aver bisogno di compiere operazioni con più di due operandi. Questo comando permette di aggiungere tanti pins di input quanti ne servono per raggiungere il numero di operandi richiesto.
	Dec pins	Annulla il comando *Inc pins*.

Bottone	Comando	Descrizione
	EN/ENO	Aggiunge i pins *enable in/enable out* al blocco selezionato. Il codice per implementare tale blocco sarà eseguito solo quando il segnale *enable in* è *true*. Il segnale *enable out* ripete semplicemente il valore di *enable in*, permettendo sia di attivare sia di disattivare la sequenza dei blocchi.
	FBD properties	Mostra alcune proprietà del blocco selezionato: - se l'oggetto è una funzione o un blocco di funzioni, mostra una tabella con le variabili input e output; - se l'oggetto è una variabile o un parametro, apre una finestra di dialogo che permette di cambiare il nome e la direzione logica (input/output).
	View source	Apre l'editor con cui l'oggetto selezionato è stato creato, e mostra il codice sorgente relativo: - Se l'oggetto è un programma, una funzione o un blocco funzione questo comando apre l'editor di codice sorgente corrispondente; - se l'oggetto è una variabile o un parametro, apre l'editor di variabili corrispondente; - se l'oggetto è una funzione standard o un operatore, il comando non apre nulla.

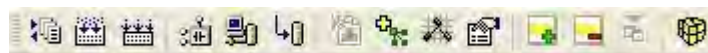
10.2.3 TOOLBAR LD

Bottone	Comando	Descrizione
	Insert parallel	Attiva la modalità di inserimento di parallele. Tutti i contatti inseriti con questa modalità verranno inseriti in parallelo con i contatti attualmente selezionati.
	Insert series	Attiva la modalità di inserimento in serie. Tutti i contatti inseriti con questa modalità verranno inseriti alla destra del contatto/blocco correntemente selezionato. Se è selezionata una connessione, il nuovo contatto sarà posizionato a metà del segmento di connessione.
	Insert contact	Inserimento di un nuovo contatto in base alla modalità selezionata (serie o parallela).
	Insert negated contact	Inserimento di un nuovo contatto negativo in base alla modalità selezionata (serie o parallela).
	Insert rising edge contact	Inserimento di un nuovo contatto rising edge in base alla modalità selezionata (serie o parallela).
	Insert falling edge contact	Inserimento di un nuovo contatto falling edge in base alla modalità selezionata (serie o parallela).
	Insert coil	Inserimento di una nuova uscita attaccata al power rail di destra.

Bottone	Comando	Descrizione
	Insert negated coil	Inserimento di una nuova uscita negata attaccata al power rail di destra.
	Insert set contact	Inserimento di un nuovo set di uscite attaccato al power rail di destra.
	Insert reset coil	Inserimento di una nuova uscita sempre spenta attaccata al power rail di destra
	Insert rising edge contact	Non ancora implementato, funziona come una normale uscita.
	Insert falling edge contact	Non ancora implementato, funziona come una normale uscita.

10.2.4 TOOLBAR SFC

Bottone	Comando	Descrizione
	New step	Inserisce un nuovo stato nel documento corrente SFC.
	Add transition	Aggiunge una nuova transizione al corrente documento SFC.
	Add jump	Aggiunge un nuovo jump al corrente documento SFC.
	Add divergent pin	Aggiunge un nuovo pin alla transizione divergente selezionata.
	Remove divergent pin	Rimuove il pin più a destra della transizione divergente selezionata.
	Add convergent pin	Aggiunge un nuovo pin alla transizione convergente selezionata.
	Remove convergent pin	Rimuove il pin più a destra della transizione convergente selezionata.
	Add simultaneous divergent pin	Aggiunge un nuovo pin alla transizione simultanea divergente selezionata.
	Remove simultaneous divergent pin	Rimuove il pin più a destra della transizione simultanea divergente selezionata.
	Add simultaneous convergent pin	Aggiunge un nuovo pin alla transizione simultanea convergente selezionata.
	Remove simultaneous convergent pin	Rimuove il pin più a destra della transizione simultanea convergente selezionata.
	Shift pin right	Aumenta la distanza tra i due pins più a destra della transizione corrente, per permettere al subnet SFC collegato al pin di destra di contenere rami divergenti.

Bottone	Comando	Descrizione
	Shift pin left	Diminuisce la distanza tra i due pins più a destra della transizione correntemente selezionata
	New action code	Permette all'utente di creare una nuova azione da associare ad uno degli stati. Premendo questo tasto, LogicLab chiede quale linguaggio si desidera utilizzare per implementare la nuova azione e apre l'editor corrispondente.
	New transition code	Permette all'utente di scrivere il codice da associare ad una delle transizioni. Premendo questo tasto, LogicLab chiede quale linguaggio si desidera utilizzare per implementare la nuova transizione e apre l'editor corrispondente.

10.2.5 TOOLBAR PROJECT

Bottone	Comando	Descrizione
	Library manager	Apre il manager library.
	Compile	Chiede se salvare le modifiche non ancora salvate, poi lancia il compilatore LogicLab.
	Recompile all	Chiede se salvare le modifiche non ancora salvate, poi lancia il compiler LogicLab per ricompilare tutto il progetto.
	Connect to the target	LogicLab cerca di stabilire una connessione con il target.
	Code download	LogicLab controlla se sono state apportate modifiche dall'ultima compilazione e in questo caso, compila il progetto. Poi invia il codice compilato al target.
	New macro	Definisce una nuova macro.
	Object browser	Apre l'object browser, che permette di navigare tra gli oggetti.
	PLC Obj properties	Mostra le proprietà e la descrizione dell' oggetto correntemente selezionato nel *Workspace*.
	Insert record	Aggiunge una nuova riga nella tabella dell'editor correntemente attivo (se è un editor PLC, alla tabella delle variabili locali; se è un editor di parametri, alla tabella di parametri, ecc.).
	Delete record	Cancella la variabile nella riga selezionata della tabella correntemente attiva.
	Generate redistributable source module	Crea un file RSM del progetto.

10.2.6 TOOLBAR NETWORK

Bottone	Comando	Descrizione
	Insert Top	Aggiunge un network vuoto nella parte alta del documento LD/FBD attivo.
	Insert Bottom	Aggiunge un network vuoto nella parte bassa del documento LD/FBD attivo.
	Insert After	Aggiunge un network vuoto dopo il network selezionato del documento LD/FBD attivo.
	Insert Before	Aggiunge un network vuoto prima del network selezionato del documento LD/FBD attivo.
	View grid	Se selezionato, mostra una griglia puntata sullo sfondo dell'editor LD/FBD.
	Auto connect	Se selezionato, abilita l'autoconnessione, che è la creazione automatica di un filo logico che collega i pins di due blocchi, una volta portati uno vicino all'altro.

10.2.7 TOOLBAR DEBUG

Bottone	Comando	Descrizione
	Debug mode	Attiva/disattiva la modalità di debug.
	Live debug mode	Attiva/disattiva la modalità di debug live.
	Set/Remove trigger	Imposta/rimuove un trigger nella linea di codice sorgente corrente.
	Graphic trigger	Imposta/rimuove un trigger grafico nella linea di codice sorgente corrente.
	Remove all triggers	Rimuove tutti i trigger.
	Trigger list	Elenca tutti i trigger.
	Set breakpoints	Imposta un breakpoint nella linea di codice sorgente corrente.
	Remove all breakpoints	Rimuove tutti i breakpoint.
	Run	Fa ripartire l'esecuzione di un programma dopo un breakpoint.
	Breakpoint list	Elenca tutti i breakpoint.
	Change current instance	Cambia la ricorrenza del blocco funzioni corrente (modalità live debug).

11. RIFERIMENTI DI LINGUAGGIO

Tutti i linguaggi di LogicLab rispettano lo standard IEC 61131-3.

- Elementi comuni.
- Instruction list (IL).
- Function block diagram (FBD).
- Ladder diagram (LD).
- Structured text (ST).
- Sequential Function Chart (SFC).

Inoltre, LogicLab aggiunge alcune estensioni:

- Puntatori.
- Macro.

11.1 ELEMENTI COMUNI

Gli elementi comuni testuali e grafici sono mezzi che sono comuni a tutti i linguaggi di programmazione di controlli programmabili specificati dallo standard IEC 61131-3.

Nota: la definizione e la modifica della maggior parte degli elementi comuni (variabili, elementi strutturati, definizioni di blocchi funzione, ecc.) sono gestiti da LogicLab attraverso editor, moduli e tabelle. LogicLab non permette di modificare direttamente il codice sorgente relativo agli elementi comuni menzionati qui sopra. I paragrafi seguenti spiegano gli elementi dei linguaggi. Per gestire correttamente gli elementi comuni, fare riferimento alla guida utente di LogicLab.

11.1.1 ELEMENTI BASE

11.1.1.1 SET DI CARATTERI

I documenti e gli elementi testuali dei linguaggi grafici sono scritti usando il set di caratteri ASCII standard.

11.1.1.2 COMMENTI

I commenti dell'utente sono delimitati all'inizio e alla fine dalle combinazioni di caratteri speciali "(*" and "*)" . I commenti sono permessi in qualunque parte del programma, e non hanno alcun significato semantico o sintattico in nessuno dei linguaggi definiti da questo standard.

L'uso di commenti nidificati, per esempio `(*(*NESTED*)*)` è segnalato come errore.

11.1.2 TIPI DI DATI ELEMENTARI

Un numero di tipi di dati elementari (ovvero predefiniti) sono resi disponibili da LogicLab, tutti compatibili con lo standard IEC 61131-3.

Nella tabella seguente sono descritti i tipi di dati elementari; la parola chiave per ciascun tipo di dato, il numero di bit per elemento di dati e la gamma di valori per ciascun tipo di dato.

Keyword	Data type	Bit	Range
BOOL	Boolean	See note	0 to 1
SINT	Short integer	8	-128 to 127
USINT	Unsigned short integer	8	0 to 255
INT	Integer	16	-32768 to 32767

Keyword	Data type	Bit	Range
UINT	Unsigned integer	16	0 to 65536
DINT	Double integer	32	-2^{31} to $2^{31}-1$
UDINT	Unsigned long integer	32	0 to 2^{32}
BYTE	Bit string of length 8	8	—
WORD	Bit string of length 16	16	—
DWORD	Bit string of length 32	32	—
REAL	Real number	32	-3.40E+38 to +3.40E+38
STRING	String of characters	-	-

Nota: la reale implementazione del tipo di dato BOOL dipende dal processore del target device, per esempio, è lungo 1 bit per i device che hanno un'area bit-indirizzabile.

11.1.3 TIPI DI DATO DERIVATI

I tipi di dato derivati possono essere dichiarati usando la costruzione TYPE…END_TYPE. Questi tipi di dati derivati possono essere utilizzati nelle dichiarazioni delle variabili, in aggiunta ai tipi di dati elementari.

Sia le variabili ad elemento singolo che gli elementi delle variabili multi-elemento, che sono descritti come tipi di dati derivati, possono essere utilizzati dovunque una variabile di questo tipo può essere utilizzata.

11.1.3.1 TYPEDEF

Lo scopo dei typedef è di assegnare nomi alternativi a tipi esistenti. Non esiste nessuna differenza tra un typedef e il suo tipo di appartenenza, tranne il nome.

I typedef possono essere dichiarati utilizzando la sintassi seguente:

```
TYPE
<enumerated data type name> : <parent type name>;
END_TYPE
```

Per esempio, considerare la dichiarazione seguente, mappare il nome LONGWORD di tipo DWORD dello standard IEC 61131-3:

```
TYPE
longword : DWORD;
END_TYPE
```

11.1.3.2 TIPI DI DATO ENUMERATIVI

Una dichiarazione di tipo di dato enumerativo stabilisce che il valore di ogni elemento di questo tipo può essere solo uno dei valori dati nella lista associata di identificatori. La lista degli enumerativi definisce un set ordinato di valori enumerati, che partono il primo identificatore della lista e finiscono con l'ultimo.

I tipi di dato enumerativi possono essere dichiarati con la seguente sintassi:

```
TYPE
<enumerated data type name> : ( <enumeration list> );
END_TYPE
```

Per esempio, considerare la seguente dichiarazione di due tipi di dati enumerativi.

Notare che, quando non è assegnato nessun valore esplicito ad un identificatore nella lista degli enumerativi, il suo valore equivale al valore assegnato all'identificatore precedente aumentato di uno.

```
TYPE
enum1: (
      val1, (* the value of val1 is 0 *)
      val2,       (* the value of val1 is 1 *)
      val3  (* the value of val1 is 2 *)
);
enum2: (
      k := -11,
      i := 0,
      j,          (* the value of j is ( i + 1 ) = 1  *)
      l := 5
);
END_TYPE
```

Diversi dati enumerativi possono utilizzare gli stessi identificatori. Per essere identificati unicamente quando utilizzati in un contesto particolare, gli enumerativi letterali possono essere qualificati con un prefisso che consiste nel nome del tipo di dato associato e il segno #.

11.1.3.3 SUBRANGE

Una dichiarazione di tipo subrange specifica che il valore di qualsiasi elemento di dato è incluso tra i limiti massimi e minimi specificati.

I subrange possono essere dichiarati con la sintassi seguente:

```
TYPE
<subrange name> : <parent type name> ( <lower limit>..<upper limit> );
END_TYPE
```

Per un esempio concreto considerare la dichiarazione seguente:

```
TYPE
int_0_to_100 : INT (0..100);
END_TYPE
```

11.1.3.4 STRUTTURE

Una dichiarazione STRUCT specifica che gli elementi di dati di questo tipo contengono sottoelementi di tipi specifici cui si può avere accesso con nomi specifici.

Le strutture possono essere dichiarate utilizzando la sintassi seguente:

```
TYPE
<structured type name> : STRUCT
      <declaration of stucture elements>
END_STRUCT;
END_TYPE
```

Considerare, per esempio, la dichiarazione seguente:

```
TYPE
structure1 : STRUCT
      elem1 : USINT;
      elem2 : USINT;
```

```
        elem3 : INT;
        elem3 : REAL;
    END_STRUCT;
END_TYPE
```

11.1.4 COSTANTI LETTERALI

11.1.4.1 COSTANTI LETTERALI NUMERICHE

La rappresentazione esterna di dati nei vari linguaggi di programmazione di controlli a logica programmabile consiste in letterali numerici.

Ci sono due classi di letterali numerici: letterali interi e letterali reali. Un letterale numerico è definito come un numero decimale o un numero con base.

I letterali decimali sono rappresentati con una connotazione decimale convenzionale. I letterali reali sono distinti grazie alla presenza della virgola decimale. Un esponente indica la potenza intera di dieci con cui si deve moltiplicare il numero precedente per ottenere il valore rappresentato. I letterali decimali e i loro esponenti possono essere preceduti da un segno (+ o -).

I letterali interi possono essere rappresentati in base 2, 8 o 16. La base è in notazione decimale. Per la base 16, è usato un set esteso di cifre che consistono nelle lettere da A a F, con il rispettivo significato convenzionale di decimale da 10 a 15. I numeri con base non contengono nessun segno (+ o -).

I dati Booleani sono rappresentati tramite FALSE o TRUE.

I caratteri e gli esempi dei letterali numerici sono riportati nella tabella sottostante.

Descrizione	Esempio
Letterali interi	-12 0 123 +986
Letterali reali	-12.0 0.0 0.4560
Letterali reali con esponente	-1.34E-12 or -1.34e-12 1.0E+6 or 1.0e+6 1.234E6 or 1.234e6
Letterale a base 2	2#11111111 (256 decimali) 2#11100000 (240 decimali)
Letterale a base 8	8#377 (256 decimali) 8#340 (240 decimali)
Letterale a base 16	16#FF or 16#ff (256 decimali) 16#E0 or 16#e0 (240 decimali)
Boolean FALSE e TRUE	FALSE TRUE

11.1.4.2 COSTANTI LETTERALI STRINGA

Una costante letterale stringa è una sequenza di zero o più caratteri che iniziano e terminano con il carattere apice singolo (‘).

Le combinazioni di tre caratteri del simbolo del dollaro ($) seguito da due cifre esadecimali devono essere interpretate come la rappresentazione esadecimale del codice a otto bit del carattere.

Esempio	Spiegazione
''	Stringa vuota (lunghezza zero)
'A'	Stringa di lunghezza uno contenente un singolo carattere A
' '	Stringa di lunghezza uno contenente il carattere *space*.
'$''	Stringa di lunghezza uno contenente il carattere *apice singolo*
'"'	Stringa di lunghezza uno contenente il carattere *virgolette*
'RL'	Stringa di lunghezza due contenente i carattere CR e LF
'$0A'	Stringa di lunghezza uno contenente il carattere LF

Le combinazioni di due caratteri che iniziano con il simbolo del dollaro devono essere interpretate, quando si verificano in stringhe di caratteri, come indicato nella tabella seguente.

Combinazione	Interpretazione
$$	Segno del dollaro
$'	Apice singolo
$L or $1	Linea di alimentazione
$N or $n	Nuova linea
$P or $p	Modulo di alimentazione (pagina)
$R or $r	Ritorno a capo
$T or $t	Tab

11.1.5 VARIABILI

11.1.5.1 INTRODUZIONE

Le variabili offrono mezzi di identificazione di oggetti di dati il cui contenuto può cambiare, per esempio i dati associati agli input e agli output o alla memoria del controllore programmabile. Una variabile deve essere dichiarata come un dato di tipo elementare. Le variabili possono essere rappresentate simbolicamente, o altrimenti in una maniera che rappresenti direttamente l'associazione dei dati con la loro posizione logica o fisica nell'input, output o memoria del controllore programmabile.

Ciascuna POU (per esempio, ciascun programma,funzione o blocco funzione) contiene almeno una sezione dichiarativa iniziale, che consiste in uno o più elementi strutturali, che specificano i tipi (e, se necessario, la posizione fisica o logica) delle variabili usate dalla POU. Questa parte dichiarativa inizia con una delle keyword VAR,VAR_INPUT o VAR_OUTPUT, opzionalmente seguita da uno o più qualificatori (RETAIN, o CONSTANT), seguita da una o più dichiarazioni separate da semicolonne e terminate dalla keyword END_VAR. Una dichiarazione può anche specificare un'inizializzazione per la variabile dichiarata, quando un controllore programmabile supporta la dichiarazione dell'utente di valori iniziali per variabili.

11.1.5.2 STRUTTURARE ELEMENTI

La dichiarazione di una variabile deve seguire la seguente sintassi:

```
KEYWORD [RETAIN] [CONSTANT]
  Declaration 1
  Declaration 2
```

```
...
 Declaration N
END_VAR
```

11.1.5.3 PAROLE CHIAVE E AMBITO

Parola chiave	Impiego della variabile
VAR	Interna all'unità di organizzazione.
VAR_INPUT	Fornita esternamente.
VAR_OUTPUT	Fornita dall'unità di organizzazione ad entità esterne.
VAR_IN_OUT	Fornita da entità esterne, può essere modificata all'interno dell'unità di organizzazione.
VAR_EXTERNAL	Fornita in base alla configurazione via VAR_GLOBAL, può essere modificata all'interno dell'unità di organizzazione.
VAR_GLOBAL	Dichiarazione di una variabile globale.

La visibilità delle dichiarazioni contenute negli elementi di struttura sono locali alla POU in cui la dichiarazione si trova.

Ovvero, le variabili dichiarate non sono accessibili alle altre POU tranne che per passaggio esplicito di parametri tramite variabili che sono state dichiarate input o output di queste unità.

L'eccezione alla regola è il caso della dichiarazione di variabili globali. Questo tipo di variabile è accessibile ad una POU tramite una dichiarazione di VAR_EXTERNAL. Il tipo di variabile dichiarata in una VAR_EXTERNAL deve accordarsi con il tipo dichiarato nel blocco VAR_GLOBAL.

Si verifica un errore se:

- una POU qualsiasi cerca di modificare il valore di una variabile che è stata dichiarata CONSTANT;
- una variabile dichiarata VAR_GLOBAL CONSTANT in un elemento di configurazione o in una POU ("elemento di contenimento") è utilizzata in una dichiarazione VAR_EXTERNAL (senza la caratteristica CONSTANT) di un elemento qualsiasi contenuto all'interno dell'elemento di contenimento.

11.1.5.4 QUALIFICATORE

Qualificatore	Descrizione
CONST	L'attributo CONST indica che le variabili all'interno degli elementi di struttura sono costanti, ovvero hanno un valore costante, che non può essere modificato una volta che il progetto PLC è stato compilato.
RETAIN	L'attributo RETAIN indica che le variabili all'interno degli elementi di struttura sono ritentive, ovvero mantengono il loro valore anche dopo che il target device è resettato o disattivato.

11.1.5.5 VARIABILI SCALARI E ARRAY

Una variabile scalare rappresenta un singolo elemento di dato di tipi elementari o di tipi di dati derivati.

Un array è una collezione di elementi di dati dello stesso tipo; per avere accesso ad un singolo elemento dell'array, deve essere usato un indice chiuso tra parentesi quadre. Gli indici possono essere sia letterali interi sia variabili ad elemento singolo.

Per rappresentare facilmente le matrici di dati, gli array possono essere multi-dimensionali; in questo caso, è richiesto un indice composto, un indice per dimensione, separati da virgole. Il numero massimo di dimensioni permesse nella definizione di un array è tre.

11.1.5.6 SINTASSI DI DICHIARAZIONE

Le variabili devono essere dichiarate all'interno degli elementi di struttura, usando la sintassi seguente:

```
VarName1 : Typename1 [ := InitialVal1 ];
VarName2 AT Location2 : Typename2 [ := InitialVal2 ];
VarName3 : ARRAY [ 0..N ] OF Typename3;
```

dove:

Parola chiave	Descrizione
`VarNameX`	Identificatore di variabile, che consiste in una stringa di caratteri alfanumerici, di lunghezza 1 o superiore. E' usato per la rappresentazione simbolica delle variabili.
`TypenameX`	Tipo di dato della variabile, selezionato da tipi di dati elementari.
`InitialValX`	Il valore che la variabile assume dopo il reset del target.
`LocationX`	Vedere il paragrafo successivo.
`N`	Indice dell'ultimo elemento, con array avente lunghezza `N+1`.

11.1.5.7 INDIRIZZO LOGICO

Le variabili possono essere rappresentate simbolicamente, ovvero è possibile accedere ad esse attraverso il loro identificatore, o, altrimenti, in un modo che rappresenta direttamente l'associazione dell'elemento di dato con la posizione logica o fisica nell'input, output o struttura di memoria del controller programmabile.

La rappresentazione diretta di una variabile a singolo elemento è fornita da un simbolo speciale formato dalla concatenazione del segno percentuale "%", un prefisso di posizionamento e uno di dimensione e uno o due interi senza segno, separati da punti (`.`).

```
%location.size.index.index
```

1) posizione:

il prefisso di posizione deve essere uno dei seguenti:

Prefisso di posizione	Descrizione
`I`	Posizione Input
`Q`	Posizione Output
`M`	Posizione Memory

2) dimensione:

il prefisso di dimensione deve essere uno dei seguenti:

Prefisso di dimensione	Descrizione
X	Bit singolo
B	Byte (8 bits)
W	Word (16 bits)
D	Double word (32 bits)

3) index.index:

questa sequenza di interi senza segno, separati da punti, specifica la posizione della variabile nell'area specificata dal prefisso di posizione.

Esempio:

Rappresentazione diretta	Descrizione
%MW4.6	Word che parte dal primo byte del settimo elemento di memoria del blocco di dati 4.
%IX0.4	Primo bit del primo byte del quinto elemento del set input 0.

Notare che la posizione assoluta dipende dalla dimensione degli elementi del blocco di dati, non dal prefisso di dimensione. Infatti, %MW4.6 e %IX0.4 iniziano dallo stesso byte in memoria, ma il primo porta ad un'area che è 16 bits più corta dell'ultimo.

Per utenti avanzati: se l'indice consiste solo in un intero (senza punti), allora perde qualsiasi riferimento al blocco di dati, e porta direttamente al byte nella memoria che ha il valore dell'indice come suo indirizzo assoluto.

Rappresentazione diretta	Descrizione
%MW4.6	Word che parte dal primo byte del settimo elemento del blocco di dati 4 in memoria.
%MW4	Word che parte dal byte 4 in memoria

Esempio

```
VAR [RETAIN] [CONSTANT]
    XQuote : DINT;     Enabling : BOOL := FALSE;
    TorqueCurrent AT %MW4.32 : INT;
    Counters : ARRAY [ 0 .. 9 ] OF UINT;
Limits: ARRAY [0..3, 0..9]
END_VAR
```

- La variabile XQuote è lunga 32 bit, ed è automaticamente posizionata dal compilatore LogicLab.
- La variabile Enabling è inizializzata su FALSE dopo il reset del target
- La variabile TorqueCurrent è posizionata nell'area di memoria del target device, utilizza 16 bits partendo dal primo byte del trentatreesimo elemento del blocco dati 4.
- La variabile Counters è un array di dieci variabili indipendenti di tipo intero senza segno.

11.1.5.8 DICHIARARE VARIABILI IN LOGICLAB

Qualsiasi linguaggio PLC sia in uso, LogicLab permette di non tener conto della sintassi riportata qui sopra, poiché fornisce l'editor di variabili locali, l'editor di variabili globali e l'editor dei parametri, che presentano un'interfaccia semplice per dichiarare tutti i tipi di variabili.

11.1.6 PROGRAM ORGANIZATION UNITS

Le POU sono funzioni, blocchi funzione e programmi. Queste POU possono essere consegnate dal produttore o programmate dall'utente attraverso i mezzi definiti in questa parte dello standard.

Le POU non sono ricorsive; ovvero, richiamare una POU non causa il richiamo di un' altra POU dello stesso tipo.

11.1.6.1 FUNZIONI

Introduzione

Per gli scopi dei linguaggi di programmazione del controller programmabile, una funzione è definita come una POU che, quando viene eseguita, produce esattamente un elemento di dato, che è considerato il risultato della funzione.

Le funzioni non contengono informazioni di stato interne, ovvero il richiamo di una funzione con gli stessi argomenti (variabili input VAR_INPUT e variabili in-out VAR_IN_OUT) produce sempre gli stessi valori (variabili output VAR_OUTPUT , variabili in-out VAR_IN_OUT e risultato della funzione).

Sintassi per la dichiarazione

La dichiarazione di una funzione deve essere eseguita in questo modo:

```
FUNCTION FunctionName : RetDataType
VAR_INPUT
    declaration of input variables (see the relevant section)
END_VAR
VAR
    declaration of local variables (see the relevant section)
END_VAR
    Function body
END_FUNCTION
```

Parola chiave	Descrizione
FunctionName	Nome della funzione dichiarata.
RetDataType	Tipo di dato del valore che la funzione ha riportato..
Function body	Specifica le operazioni che devono essere eseguite sulle variabili input per assegnare valori dipendenti dalla semantica della funzione ad una variabile con lo stesso nome della funzione, che rappresenta il risultato della funzione stessa. Può essere scritto in un qualsiasi linguaggio supportato da LogicLab.

Dichiarare funzioni in LogicLab

Qualunque sia il linguaggio PLC in uso, LogicLab permette di ignorare la sintassi riportata qui sopra, poiché fornisce un'interfaccia semplice per l'uso delle funzioni.

11.1.6.2 BLOCCHI FUNZIONI

Introduzione

Per gli scopi dei linguaggi di programmazione del controller programmabile, un blocco funzione è una POU che, quando viene eseguita, riporta uno o più valori. Possono essere create ricorrenze multiple, nominate (copie) di un blocco funzione. Ogni ricorrenza ha un identifier associato (il nome della ricorrenza) e una struttura di dati contenente il suo input, output e variabili interne. Tutti i valori delle variabili output e le variabili interne necessarie di questa struttura di dati si mantengono da un'esecuzione del blocco funzione all'altra; per questo motivo, il richiamo di un blocco funzione con gli stessi argomenti (variabili input) non produce sempre gli stessi valori di output.

Solo le variabili input e output sono accessibili al di fuori di una ricorrenza di un blocco funzione, ovvero, le variabili interne del blocco funzione sono nascoste all'utente del blocco funzione.

Per eseguire le sue operazioni, un blocco funzione ha bisogno di essere richiamato da un'altra POU. Il richiamo dipende dal linguaggio specifico del modulo che chiama il blocco funzione. Il raggio di una ricorrenza di un blocco funzione è locale alla POU nella quale è rappresentata.

Sintassi per la dichiarazione

La dichiarazione di una funzione deve essere eseguita come segue:

```
FUNCTION_BLOCK FunctionBlockName
 VAR_INPUT
     declaration of input variables (see the relevant section)
 END_VAR
 VAR_OUTPUT
     declaration of output variables
 END_VAR
 VAR_EXTERNAL
     declaration of external variables
 END_VAR
 VAR
     declaration of local variables
 END_VAR
    Function block body
END_FUNCTION_BLOCK
```

Parola chiave	Descrizione
FunctionBlockName	Nome del blocco funzione dichiarato (notare: nome del modello, non delle sue ricorrenze).
VAR_EXTERNAL .. END_VAR	Un blocco funzione può accedere alle variabili globali solo se sono elencate in un elemento di struttura VAR_EXTERNAL. Le variabili passate al FB tramite un costrutto VAR_EXTERNAL possono essere modificate all'interno del FB.
Function block body	Specifica le operazioni da eseguire sulle variabili input per assegnare valori alle variabili output –dipendenti dalla semantica del blocco funzione e dal valore delle variabili interne. Può essere scritta in uno qualunque dei linguaggi supportati da LogicLab.

Dichiarare funzioni in LogicLab

Qualunque linguaggio PLC sia in uso, LogicLab permette di non tener conto della sintassi riportata qui sopra poichè fornisce un'interfaccia semplice per usare i blocchi funzione.

11.1.6.3 PROGRAMMI

Introduzione

Un programma è definito in IEC 61131-3 come un "assemblamento logico di tutti gli elementi e costrutti del linguaggio di programmazione necessari per l'elaborazione del segnale intenzionale richiesto da un sistema di controlli a logica programmabile per il controllo di una macchina o di un processo".

Sintassi per la dichiarazione

La dichiarazione di un programma deve essere eseguita come segue:

```
PROGRAM < program name>
    Declaration of variables (see the relevant section)
    Program body
END_PROGRAM
```

Parola chiave	Descrizione
`Program Name`	Nome del programma dichiarato.
`Program body`	Specifica le operazioni da eseguire per ottenere l'elaborazione del segnale intenzionale. Può essere scritta in uno qualsiasi dei linguaggi supportati da LogicLab.

Scrivere un programma in LogicLab

Qualunque sia il linguaggio PLC in uso, LogicLab permette di non tener conto della sintassi riportata qui sopra, poiché fornisce un'interfaccia semplice per scrivere i programmi.

11.1.7 FUNZIONI STANDARD IEC 61131-3

Questo paragrafo descrive tutte le funzioni standard IEC 61131-3 disponibili in LogicLab, più altre che possono essere considerate come estensioni di LogicLab allo standard.

Queste funzioni sono comuni all'intero set di linguaggi di programmazione e possono quindi essere usate in qualunque Programmable Organization Unit (POU).

Una funzione indicata come (Ext.) può avere un numero variabile di input.

Funzioni di conversione di tipo

In base allo standard, le funzioni di conversione devono avere la forma `*_TO_**`, dove "*" è il tipo di variabile input, e "**" è il tipo della variabile di output (per esempio, `INT_TO_REAL`). LogicLab fornisce un set di funzioni di conversione che evitano allo sviluppatore di specificare il tipo della variabile di input.

TO_BOOL	
Descrizione	Conversione in BOOL (booleano)
Numero di operandi	1
Tipo di dato input	Qualsiasi tipo numerico
Tipo di dato output	BOOL

Esempi	`out := TO_BOOL( 0 ); (* out = FALSE *)`
	`out := TO_BOOL( 1 ); (* out = TRUE *)`
	`out := TO_BOOL( 1000 ); (* out = TRUE *)`

TO_SINT	
Descrizione	Conversione in SINT (intero con segno di 8 bit)
Numero di operandi	1
Tipo di dato input	Qualsiasi tipo numerico
Tipo di dato output	SINT
Esempi	`out := TO_SINT( -1 ); (* out = -1 *)` `out := TO_SINT( 16#100 ); (* out = 0 *)`

TO_USINT	
Descrizione	Conversione in USINT (intero senza segno di 8 bit)
Numero di operandi	1
Tipo di dato input	Qualsiasi tipo numerico
Tipo di dato output	USINT
Esempi	`out := TO_USINT( -1 ); (* out = 255 *)` `out := TO_USINT( 16#100 ); (* out = 0 *)`

TO_INT	
Descrizione	Conversione in INT (intero con segno di 16 bit)
Numero di operandi	1
Tipo di dato input	Qualsiasi tipo numerico
Tipo di dato output	INT
Esempi	`out := TO_INT( -1000.0 ); (* out = -1000 *)` `out := TO_INT( 16#8000 ); (* out = -32768 *)`

TO_UINT	
Descrizione	Conversione in UINT (intero senza segno di 16 bit)
Numero di operandi	1
Tipo di dato input	Qualsiasi tipo numerico
Tipo di dato output	UINT
Esempi	`out := TO_UINT( 1000.0 ); (* out = 1000 *)` `out := TO_UINT( 16#8000 ); (* out = 32768 *)`

TO_DINT	
Descrizione	Conversione in DINT (intero con segno di 32 bit)
Numero di operandi	1
Tipo di dato input	Qualsiasi tipo numerico
Tipo di dato output	DINT
Esempi	`out := TO_DINT( 10.0 ); (* out = 10 *)` `out := TO_DINT( 16#FFFFFFFF ); (* out = -1 *)`

TO_UDINT	
Descrizione	Conversione in UDINT (intero senza segno di 32 bit)
Numero di operandi	1
Tipo di dato input	Qualsiasi tipo numerico
Tipo di dato output	UDINT
Esempi	`out := TO_UDINT( 10.0 ); (* out = 10 *)` `out := TO_UDINT( 16#FFFFFFFF ); (* out = 4294967295 *)`

TO_BYTE	
Descrizione	Conversione in BYTE (8 bit)
Numero di operandi	1
Tipo di dato input	Qualsiasi tipo numerico
Tipo di dato output	BYTE
Esempi	`out := TO_BYTE( -1 ); (* out = 16#FF *)` `out := TO_BYTE( 16#100 ); (* out = 16#00 *)`

TO_WORD	
Descrizione	Conversione in WORD (16 bit)
Numero di operandi	1
Tipo di dato input	Qualsiasi tipo numerico
Tipo di dato output	WORD
Esempi	`out := TO_WORD( 1000.0 ); (* out = 16#03E8 *)` `out := TO_WORD( -32768 ); (* out = 16#8000 *)`

TO_DWORD	
Descrizione	Conversione in DWORD (32 bit)
Numero di operandi	1
Tipo di dato input	Qualsiasi tipo numerico
Tipo di dato output	DWORD
Esempi	`out := TO_DWORD( 10.0 ); (* out = 16#0000000A *)` `out := TO_DWORD( -1 ); (* out = 16#FFFFFFFF *)`

TO_REAL	
Descrizione	Conversione in REAL (virgola mobile 32 bit)
Numero di operandi	1
Tipo di dato input	Qualsiasi tipo numerico
Tipo di dato output	REAL
Esempi	`out := TO_REAL( -1000 ); (* out = -1000.0 *)` `out := TO_REAL( 16#8000 ); (* out = -32768.0 *)`

TO_LREAL	
Descrizione	Conversione in LREAL (virgola mobile 64 bit)
Numero di operandi	1
Tipo di dato input	Qualsiasi tipo numerico
Tipo di dato output	LREAL
Esempi	`out := TO_LREAL( -1000 ); (* out = -1000.0 *)` `out := TO_LREAL( 16#8000 ); (* out = -32768.0 *)`

Funzioni numeriche

La disponibilità delle seguenti funzioni dipende dal sistema target. Fare riferimento al proprio fornitore hardware per maggiorni informazioni.

ABS	
Descrizione	Valore assoluto. Calcola il valore assoluto dell'input #0
Numero di operandi	1
Tipo di dato input	Qualsiasi tipo numerico
Tipo di dato output	Lo stesso dell'input

Esempi	`OUT := ABS( -5 );(* OUT = 5 *)`
	`OUT := ABS( -1.618 );(* OUT = 1.618 *)`
	`OUT := ABS( 3.141592 );(* OUT = 3.141592 *)`

SQRT	
Descrizione	Radice quadrata. Calcola la radice quadrata dell'input #0
Numero di operandi	1
Tipo di dato input	LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := SQRT( 4.0 ); (* OUT = 2.0 *)`

LN	
Descrizione	Logaritmo naturale. Calcola il logaritmo in base e di input #0
Numero di operandi	1
Tipo di dato input	LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := LN( 2.718281 ); (* OUT = 1.0 *)`

LOG	
Descrizione	Logaritmo. Calcola il logaritmo in base 10 di input #0
Numero di operandi	1
Tipo di dato input	LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := LOG( 100.0 );(* OUT = 2.0 *)`

EXP	
Descrizione	Esponente naturale. Calcola la funzione esponenziale di e alla input #0
Numero di operandi	1
Tipo di dato input	LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := EXP( 1.0 ); (* OUT ~ 2.718281 *)`

SIN	
Descrizione	Seno. Calcola il seno di input #0 espresso in radianti
Numero di operandi	1
Tipo di dato input	LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := SIN( 0.0 ); (* OUT = 0.0 *)` `OUT := SIN( 2.5 * 3.141592 ); (* OUT ~ 1.0 *)`

COS	
Descrizione	Coseno. Calcola il coseno di input #0 espresso in radianti
Numero di operandi	1
Tipo di dato input	LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := COS( 0.0 ); (* OUT = 1.0 *)` `OUT := COS( -3.141592 ); (* OUT ~ -1.0 *)`

TAN	
Descrizione	Tangente. Calcola la tangente di input #0 espressa in radianti
Numero di operandi	1
Tipo di dato input	LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := TAN( 0.0 ); (* OUT = 0.0 *)` `OUT := TAN( 3.141592 / 4.0 ); (* OUT ~ 1.0 *)`

ASIN	
Descrizione	Arcsin. Calcola l'arcsin di input #0 espresso in radianti
Numero di operandi	1
Tipo di dato input	LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := ASIN( 0.0 ); (* OUT = 0.0 *)` `OUT := ASIN( 1.0 ); (* OUT = PI / 2 *)`

ACOS	
Descrizione	Arcoseno. Calcola l'arcoseno di input #0; il risultato è espresso in radianti
Numero di operandi	1
Tipo di dato input	LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := ACOS( 1.0 ); (* OUT = 0.0 *)` `OUT := ACOS( -1.0 ); (* OUT = PI *)`

ATAN	
Descrizione	Arcotangente. Calcola l'arcotangente di input #0; il risultato è espresso in radianti
Numero di operandi	1
Tipo di dato input	LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := ATAN( 0.0 ); (* OUT = 0.0 *)` `OUT := ATAN( 1.0 ); (* OUT = PI / 4 *)`

ADD	
Descrizione	Somma. Calcola la somma dei due input.
Numero di operandi	2
Tipo di dato input	Qualsiasi tipo numerico, Qualsiasi tipo numerico
Tipo di dato output	Lo stesso degli input
Esempi	`OUT := ADD( 20, 40 ); (* OUT = 60 *)`

MUL	
Descrizione	Moltiplicazione. Moltiplica i due input.
Numero di operandi	2
Tipo di dato input	Qualsiasi tipo numerico, Qualsiasi tipo numerico
Tipo di dato output	Lo stesso degli input
Esempi	`OUT := MUL( 10, 10 ); (* OUT = 100 *)`

SUB	
Descrizione	Sottrazione. Sottrae input #1 da input #0
Numero di operandi	2
Tipo di dato input	Qualsiasi tipo numerico, Qualsiasi tipo numerico
Tipo di dato output	Lo stesso degli input
Esempi	`OUT := SUB( 10, 3 ); (* OUT = 7 *)`

DIV	
Descrizione	Divisione. Divide input #0 con input #1
Numero di operandi	2
Tipo di dato input	Qualsiasi tipo numerico, Qualsiasi tipo numerico
Tipo di dato output	Lo stesso degli input
Esempi	`OUT := DIV( 20, 2 ); (* OUT = 10 *)`

MOD	
Descrizione	Resto. Calcola il resto della divisione intera di input #0 diviso input #1
Numero di operandi	2
Tipo di dato input	Qualsiasi tipo numerico, Qualsiasi tipo numerico
Tipo di dato output	Lo stesso degli input
Esempi	`OUT := MOD( 10, 3 ); (* OUT = 1 *)`

POW	
Descrizione	Potenza. Elevamento di base alla potenza Expo
Numero di operandi	2
Tipo di dato input	LREAL dove disponibile, altrimenti REAL LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := POW( 2.0, 3.0 ); (* OUT = 8.0 *)` `OUT := POW( -1.0, 5.0 ); (* OUT = -1.0 *)`

ATAN2*	
Descrizione	Arcotangente (con 2 parametri). Calcola l'arcotangente di Y/X; il risultato è espresso in radianti
Numero di operandi	2
Tipo di dato input	LREAL dove disponibile, altrimenti REAL; LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL

Esempi	`OUT := ATAN2( 0.0, 1.0 ); (* OUT = 0.0 *)`
	`OUT := ATAN2( 1.0, 1.0 ); (* OUT = PI / 4 *)`
	`OUT := ATAN2( -1.0, -1.0 ); (* OUT = ( -3/4 ) * PI *)`
	`OUT := ATAN2( 1.0, 0.0 ); (* OUT = PI / 2 *)`

SINH*	
Descrizione	Seno iperbolico. Calcola il seno iperbolico dell'input #0
Numero di operandi	1
Tipo di dato input	LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := SINH( 0.0 ); (* OUT = 0.0 *)`

COSH*	
Descrizione	Coseno iperbolico. Calcola il coseno iperbolico dell'input #0
Numero di operandi	1
Tipo di dato input	LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := COSH( 0.0 ); (* OUT = 1.0 *)`

TANH*	
Descrizione	Tangente iperbolica. Calcola la tangente iperbolica dell'input #0
Numero di operandi	1
Tipo di dato input	LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := TANH( 0.0 ); (* OUT = 0.0 *)`

CEIL*	
Descrizione	Arrotondamento all'intero superiore. Restituisce il più piccolo intero che è maggiore o uguale all'input #0
Numero di operandi	1
Tipo di dato input	LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := CEIL( 1.95 ); (* OUT = 2.0 *)` `OUT := CEIL( -1.27 ); (* OUT = -1.0 *)`

FLOOR*	
Descrizione	Arrotondamento all'intero inferiore. Restituisce l'intero più grande che è minore o uguale all'input #0
Numero di operandi	1
Tipo di dato input	LREAL dove disponibile, altrimenti REAL
Tipo di dato output	LREAL dove disponibile, altrimenti REAL
Esempi	`OUT := FLOOR( 1.95 ); (* OUT = 1.0 *)` `OUT := FLOOR( -1.27 ); (* OUT = -2.0 *)`

*: funzioni fornite come estensione allo standard IEC 61131-3.

Funzioni stringhe di bit

SHL	
Descrizione	Shift a sinistra di input#0 per il numero di bit indicato da input #1 con riempimento di zero a destra.
Numero di operandi	2
Tipo di dato input	Qualsiasi tipo numerico, Qualsiasi tipo numerico
Tipo di dato output	Lo stesso di input #0
Esempi	`OUT := SHL( IN := 16#1000CAFE, 16 );` `(* OUT = 16#CAFE0000 *)`

SHR	
Descrizione	Shift a destra di input #0 per il numero di bit indicato da input #1 con riempimento di zero a sinistra.
Numero di operandi	2
Tipo di dato input	Qualsiasi tipo numerico, Qualsiasi tipo numerico
Tipo di dato output	Lo stesso di input #0
Esempi	`OUT := SHR( IN := 16#1000CAFE, 24 );` `(* OUT = 16#00000010 *)`

ROL	
Descrizione	Shift circolare a sinistra di input #0 per il numero di bit indicato da input #1.
Numero di operandi	2
Tipo di dato input	Qualsiasi tipo numerico, Qualsiasi tipo numerico
Tipo di dato output	Lo stesso di input #0
Esempi	`OUT := ROL( IN := 16#1000CAFE, 4 );` `(* OUT = 16#000CAFE1 *)`

ROR	
Descrizione	Shift circolare a destra di input #0 per il numero di bit indicato da input #1.
Numero di operandi	2
Tipo di dato input	Qualsiasi tipo numerico, Qualsiasi tipo numerico
Tipo di dato output	Lo stesso di input #0
Esempi	`OUT := ROR( IN := 16#1000CAFE, 16 );` `(* OUT = 16#CAFE1000 *)`

AND	
Descrizione	AND logico se input #0 e input #1 sono entrambi BOOL, altrimenti AND bit a bit.
Numero di operandi	2
Tipo di dato input	Tutti i tipi tranne il tipo STRING, Tutti i tipi tranne il tipo STRING
Tipo di dato output	Lo stesso degli input
Esempi	`OUT := TRUE AND FALSE;   (* OUT = FALSE *)` `OUT := 16#1234 AND 16#5678;   (* OUT = 16#1230 *)`

OR	
Descrizione	OR logico se input #0 e input #1 sono entrambi BOOL, altrimenti OR bit a bit.
Numero di operandi	2
Tipo di dato input	Tutti i tipi tranne il tipo STRING, Tutti i tipi tranne il tipo STRING
Tipo di dato output	Lo stesso degli input
Esempi	`OUT := TRUE OR FALSE; (* OUT = FALSE *)` `OUT := 16#1234 OR 16#5678;(* OUT = 16#567C *)`

XOR	
Descrizione	XOR logico se input #0 e input #1 sono entrambi BOOL, altrimenti XOR bit a bit.
Numero di operandi	2
Tipo di dato input	Tutti i tipi tranne il tipo STRING, Tutti i tipi tranne il tipo STRING
Tipo di dato output	Lo stesso degli input
Esempi	`OUT := TRUE OR FALSE; (* OUT = TRUE *)` `OUT := 16#1234 OR 16#5678; (* OUT = 16#444C *)`

NOT	
Descrizione	Negazione logica se l'input è BOOL, altrimenti NOT bit a bit.
Numero di operandi	1
Tipo di dato input	Tutti i tipi tranne il tipo STRING
Tipo di dato output	Lo stesso degli input
Esempi	`OUT := NOT FALSE; (* OUT = TRUE *)` `OUT := NOT 16#1234;(* OUT = 16#EDCB *)`

Funzioni di selezione

SEL	
Descrizione	Selezione tra due input
Numero di operandi	3
Tipo di dato input	BOOL, Qualsiasi, Qualsiasi
Tipo di dato output	Lo stesso dell'input selezionato
Esempi	`OUT := SEL( G := FALSE, IN0 := X, IN1 := 5 );` `(* OUT = X *)`

MAX	
Descrizione	Selezione del valore massimo
Numero di operandi	2, estendibile
Tipo di dato input	Qualsiasi tipo numerico, Qualsiasi tipo numerico, ..., Qualsiasi tipo numerico
Tipo di dato output	Stesso tipo dell'input massimo
Esempi	`OUT := MAX( -8, 120, -1000 ); (* OUT = 120 *)`

MIN	
Descrizione	Selezione del valore minimo
Numero di operandi	2, estendibile
Tipo di dato input	Qualsiasi tipo numerico, Qualsiasi tipo numerico, ..., Qualsiasi tipo numerico
Tipo di dato output	Stesso tipo dell'input minimo
Esempi	`OUT := MIN( -8, 120, -1000 ); (* OUT = -1000 *)`

LIMIT	
Descrizione	Limita input #0 ad essere uguale o maggiore di input#1, e uguale o minore di input #2.
Numero di operandi	3
Tipo di dato input	Qualsiasi tipo numerico, Qualsiasi tipo numerico, Qualsiasi tipo numerico
Tipo di dato output	Lo stesso degli input

Esempi	`OUT := LIMIT( IN := 4, MN := 0, MX := 5 ); (* OUT = 4 *)`
	`OUT := LIMIT( IN := 88, MN := 0, MX := 5 );(* OUT = 5 *)`
	`OUT := LIMIT( IN := -1, MN := 0, MX := 5 );(* OUT = 0 *)`

MUX	
Descrizione	Multiplexer. Seleziona uno degli N input in base al valore dell'input K
Numero di operandi	3, estendibile
Tipo di dato input	Qualsiasi tipo numerico, Qualsiasi tipo numerico, ..., Qualsiasi tipo numerico
Tipo di dato output	Lo stesso dell'input selezionato
Esempi	`OUT := MUX( 0, A, B, C ); (* OUT = A *)`

Funzioni di comparazione

Questo tipo di funzioni può essere utilizzato per comparare stringhe se la funzionalità è supportata dal target.

GT	
Descrizione	Più grande di. Restituisce TRUE se input #0 > di input #1, altrimenti FALSE.
Numero di operandi	2
Tipo di dato input	Tutti i tipi tranne BOOL, Tutti i tipi tranne BOOL
Tipo di dato output	BOOL
Esempi	`OUT := GT( 0, 20 ); (* OUT = FALSE *)` `OUT := GT( ‘pippo’, ‘pluto’ ); (* OUT = TRUE *)`

GE	
Descrizione	Maggiore o uguale di. Restituisce TRUE se input #0 >= input #1, altrimenti FALSE.
Numero di operandi	2
Tipo di dato input	Tutti i tipi tranne BOOL, Tutti i tipi tranne BOOL
Tipo di dato output	BOOL
Esempi	`OUT := GE( 20, 20 ); (* OUT = TRUE *)` `OUT := GE( ‘pippo’, ‘pluto’ ); (* OUT = FALSE *)`

EQ	
Descrizione	Uguale a. Restituisce TRUE se input #0 = input #1, altrimenti FALSE.
Numero di operandi	2
Tipo di dato input	Qualsiasi, Qualsiasi
Tipo di dato output	BOOL
Esempi	`OUT := EQ( TRUE, FALSE );     (* OUT = FALSE *)` `OUT := EQ( ‘pippo’, ‘pluto’ ); (* OUT = FALSE *)`

LT	
Descrizione	Minore di. Restituisce TRUE se input #0 < input #1, altrimenti FALSE.
Numero di operandi	2
Tipo di dato input	Tutti i tipi tranne BOOL, Tutti i tipi tranne BOOL
Tipo di dato output	BOOL
Esempi	`OUT := LT( 0, 20 ); (* OUT = TRUE *)` `OUT := LT( ‘pipp’, ‘pluto’ ); (* OUT = TRUE *)`

LE	
Descrizione	Minore o uguale di. Restituisce TRUE se input #0 <= input #1, altrimenti FALSE.
Numero di operandi	2
Tipo di dato input	Tutti i tipi tranne BOOL, Tutti i tipi tranne BOOL
Tipo di dato output	BOOL
Esempi	`OUT := LE( 20, 20 ); (* OUT = TRUE *)` `OUT := LE( ‘pipp’, ‘pluto’ ); (* OUT = TRUE *)`

NE	
Descrizione	Diverso da. Restituisce TRUE se input #0 != input #1, altrimenti FALSE.
Numero di operandi	2
Tipo di dato input	Qualsiasi, Qualsiasi
Tipo di dato output	BOOL
Esempi	`OUT := NE( TRUE, FALSE ); (* OUT = TRUE *)` `OUT := NE( ‘pipp’, ‘pluto’ ); (* OUT = TRUE *)`

Funzioni stringhe

La disponibilità delle seguenti funzioni dipende dal sistema target. Fare riferimento al proprio fornitore hardware per maggiori informazioni.

CONCAT	
Descrizione	Concatenazione di stringhe
Numero di operandi	2
Tipo di dato input	STRING, STRING
Tipo di dato output	STRING
Esempi	`OUT := CONCAT( ‘AB’, ‘CD’ ); (* OUT = ‘ABCD’ *)`

DELETE	
Descrizione	Cancella L caratteri di IN, a partire dal carattere in P-esima posizione
Numero di operandi	3
Tipo di dato input	STRING, UINT, UINT
Tipo di dato output	STRING
Esempi	`OUT := DELETE( IN := ‘ABXYC’, L := 2, P := 3 );` `(* OUT = ‘ABC’ *)`

FIND	
Descrizione	Indica la posizione del carattere iniziale della prima occorrenza di IN2 in IN1. Se IN2 non viene trovata, ritorna OUT := 0.
Numero di operandi	2
Tipo di dato input	STRING, STRING
Tipo di dato output	UINT
Esempi	`OUT := FIND( IN1 := ‘ABCBC’, IN2 := ‘BC’ ); (* OUT = 2 *)`

INSERT	
Descrizione	Inserisce IN2 in IN1 dopo il carattere in posizione P
Numero di operandi	3
Tipo di dato input	STRING, STRING, UINT
Tipo di dato output	STRING
Esempi	`OUT := INSERT( IN1 := 'ABC', IN2 := 'XY', P := 2 );` `(* OUT = 'ABXYC' *)`

LEFT	
Descrizione	Restituisce gli L caratteri a partire da sinistra della stringa IN
Numero di operandi	2
Tipo di dato input	STRING, UINT
Tipo di dato output	STRING
Esempi	`OUT := LEFT( IN := 'ASTR', L := 3 ); (* OUT = 'AST' *)`

LEN	
Descrizione	Lunghezza della stringa
Numero di operandi	1
Tipo di dato input	STRING
Tipo di dato output	UINT
Esempi	`OUT := LEN( 'ASTRING' ); (* OUT = 7 *)`

MID	
Descrizione	Restituisce gli L caratteri di IN, a partire dalla posizione indicata da P
Numero di operandi	3
Tipo di dato input	STRING, UINT, UINT
Tipo di dato output	STRING
Esempi	`OUT := MID( IN := 'ASTR', L := 2, P := 2 );` `(* OUT = 'ST' *)`

REPLACE	
Descrizione	Sostituisce gli L caratteri di IN1 che iniziano alla posizione P-esima con IN2
Numero di operandi	4
Tipo di dato input	STRING, STRING, UINT, UINT
Tipo di dato output	STRING
Esempi	`OUT := REPLACE( IN1 := 'ABCDE', IN2 := 'X', L := 2, P := 3 ); (* OUT = 'ABXE' *)`

RIGHT	
Descrizione	Restituisce gli L caratteri a partire da destra della stringa IN
Numero di operandi	2
Tipo di dato input	STRING, UINT
Tipo di dato output	STRING
Esempi	`OUT := RIGHT( IN := 'ASTR', L := 3 ); (* OUT = 'STR' *)`

11.2 INSTRUCTION LIST (IL)

Questa sezione definisce la semantica del linguaggio IL (Instruction List).

11.2.1 SINTASSI E SEMANTICA

11.2.1.1 SINTASSI DELLE ISTRUZIONI IL

Il codice IL è composto da una sequenza di istruzioni. Ciascuna istruzione comincia su una nuova linea e contiene un operatore con modificatori opzionali, e, se necessario per operazioni particolari, uno o più operandi separati da virgole. Gli operandi possono essere di una qualsiasi delle rappresentazioni di dati per i letterali e per le variabili.

Le istruzioni possono essere precedute da un'etichetta di identificazione seguita dai due punti (:). Possono essere inserite delle linee vuote tra le istruzioni.

Esempi

Analizziamo un segmento di codice:

```
START:
    LD %IX1 (* Push button *)
    ANDN %MX5.4 (* Not inhibited *)
    ST %QX2 (* Fan out *)
```

Gli elementi che costituiscono ciascuna istruzione sono classificati come segue:

Etichetta	Operatore [+ modificatore]	Operando	Commento
START:	LD	%IX1	(* Push button *)
	ANDN	%MX5.4	(* Not inhibited *)
	ST	%QX2	(* Fan out *)

Semantica delle istruzioni IL

- Accumulatori

  Con accumulatore, si intende che un registro contiene il valore del risultato correntemente analizzato.

- Operatori

  Se non specificato diversamente, la semantica degli operatori è:

  ```
  accumulator := accumulator OP operand
  ```

  Ovvero, il valore dell'accumulatore è rimpiazzato dal risultato prodotto dall'operazione `OP` applicata al valore corrente dello stesso accumulatore, con rispetto dell'operando. In tal caso, l'istruzione "`AND %IX1`" è interpretata come:

  ```
  accumulator := accumulator AND %IX1
  ```

e l'istruzione "GT %IW10" avrà il risultato Booleano TRUE se il valore corrente dell'accumulatore è più grande del valore dell'input word 10, e in caso contrario il risultato Booleano FALSE:

```
accumulator := accumulator GT %IW10
```

- Modificatori

Il modificatore "N" indica negazione bit a bit dell'operando.

Il modificatore parentesi sinistra "(" indica che la misurazione dell'operatore deve essere differita finchè non si incontra un operatore parentesi destra ")". La forma della sequenza di informazioni tra parentesi è mostrata qui sotto:

```
accumulator := accumulator AND (%MX1.3 OR %MX1.4)
```

Il modificatore "C" indica che l'istruzione associata può essere eseguita solo se il valore del risultato correntemente analizzato è Booleano 1 (o Booleano 0 se l'operatore è combinato con modificatore "N").

11.2.2 OPERATORI STANDARD

Gli operatori standard con i loro modificatori e operandi permessi sono elencati qui sotto.

Operatore	Modificatore	Tipi di operatori supportati: Acc_type, Op_type	Semantica
LD	N	Any, Any	Imposta l'accumulatore uguale all'operando.
ST	N	Any, Any	Situa l'accumulatore nella posizione dell'operando.
S		BOOL, BOOL	Imposta l'operando a TRUE se l'accumulatore è TRUE.
R		BOOL, BOOL	Imposta l'operando a FALSE se l'accumulatore è TRUE.
AND	N, (	Any but REAL, Any but REAL	AND logico o bit a bit
OR	N, (	Any but REAL, Any but REAL	OR logico o bit a bit
XOR	N, (	Any but REAL, Any but REAL	XOR logico o bit a bit
NOT		Any but REAL	NOT logico o bit a bit
ADD	(	Any but BOOL	Addizione
SUB	(	Any but BOOL	Sottrazione
MUL	(	Any but BOOL	Moltiplicazione
DIV	(	Any but BOOL	Divisione
MOD	(	Any but BOOL	Modulo-divisione
GT	(	Any but BOOL	Comparazione:
GE	(	Any but BOOL	Comparazione: =
EQ	(	Any but BOOL	Comparazione: =
NE	(	Any but BOOL	Comparazione:
LE	(	Any but BOOL	Comparazione:
LT	(	Any but BOOL	Comparazione:
JMP	C, N	Label	Salta all'etichetta
CAL	C, N	FB instance name	Richiama un function block

Operator	Modifiers	Supported operand types: Acc_type, Op_type	Semantics
RET	C, N		Riporta al programma, alla funzione o al blocco funzione richiamati.
)			Valuta le operazioni differite.

11.2.3 CHIAMATE A FUNZIONI E BLOCCHI DI FUNZIONE

11.2.3.1 CHIAMATE A FUNZIONI

Le funzioni (come definite nella sezione corrispondente) sono richiamate ponendo il nome della funzione nel campo dell'operatore. Questo richiamo ha la struttura seguente:

```
LD 1
MUX 5, var0, -6.5, 3.14
ST vRES
```

Notare che il primo argomento non è contenuto nella lista degli input, ma l'accumulatore è usato come il primo argomento della funzione. Argomenti addizionali (partendo dal secondo) se richiesti, sono riportati nel campo operando, separati da virgole, nell'ordine della loro dichiarazione. Per esempio, l'operatore MUX nella tabella qui sopra, si serve di cinque operandi, il primo dei quali è caricato nell'accumulatore, mentre gli altri quattro argomenti sono riportati ordinatamente dopo il nome della funzione.

Le seguenti regole vengono applicate per richiamare le funzioni

1) Le assegnazioni di argomenti VAR_INPUT possono essere vuoti, costanti, o variabili.
2) L'esecuzione di una funzione finisce raggiungendo un'istruzione RET o con la fine fisica della funzione. Quando accade ciò, la variabile output della funzione è copiata nell'accumulatore.

Richiamare Function Blocks

I blocchi funzione (come definiti nella sezione corrispondente) possono essere richiamati condizionatamente e incondizionatamente tramite l'operatore CAL. Questo richiamo avrà la struttura seguente:

```
LD A
ADD 5
ST INST5.IN1
LD 3.141592
ST INST5.IN2
CAL INST5
LD INST5.OUT1
ST vRES
LD INST5.OUT2
ST vVALID
```

Questo metodo di richiamo è equivalente ad un CAL con un elenco di argomenti, che contiene solo una variabile col nome della FB instance.

Gli argomenti Input sono passati a / gli argomenti output sono letti da FB instance tramite operazione ST/LD eseguite su operandi con la struttura seguente:

```
FBInstanceName.IO_var
```

dove

Parola chiave	Descrizione
FBInstanceName	Nome della ricorrenza che si vuole richiamare.
IO_var	Variabile input o output da scrivere / leggere.

11.3 FUNCTION BLOCK DIAGRAM (FBD)

Questa sezione definisce la semantica del linguaggio FBD.

11.3.1 RAPPRESENTAZIONE DI LINEE E BLOCCHI

Gli elementi di linguaggio grafico sono disegnati utilizzando elementi grafici o semi-grafici, come mostrato nella tabella qui sotto.

Nessuna riserva di dati o associazione con elementi di dato può essere associata all'uso dei connettori; quindi, per evitare ambiguità, non è possibile attribuire nessun identificatore ai connettori.

Aspetto	Esempio
Linee	
Incrocio di linee con connessione	
Blocchi con linee di connessione e pins sconnessi	

11.3.2 DIREZIONE DI FLUSSO NEI NETWORK

Un network è definito come un set massimo di elementi grafici interconnessi. Un'etichetta di network è delimitata a destra dai due punti (:) può essere associata ad ogni network o gruppo di network. Il raggio di un network e della sua etichetta è locale alla POU in cui il network è inserito.

I linguaggi grafici sono utilizzati per rappresentare il flusso di un quantità concettuale attraverso uno o più network che rappresentano un piano di controllo.

Vale a dire, nel caso di function block diagrams (FBD), il "Signal flow" è tipicamente usato, analogamente al flusso di segnali tra gli elementi di un sistema di elaborazione di segnali.

Il flusso di segnale in linguaggio FBD è dalla parte output (a destra) di una funzione o un blocco funzione alla parte input (a sinistra) della funzione o del blocco funzione così connessi.

11.3.3 VALUTAZIONE DEI NETWORK

11.3.3.1 ORDINE DI VALUTAZIONE DEI NETWORK

L'ordine in cui i network e i loro elementi sono analizzati non è necessariamente lo stesso in cui sono etichettati o visualizzati. Quando il corpo di una POU consiste in uno o più network, i risultati dell' analisi del network all'interno del suddetto corpo sono funzionalmente equivalenti all'osservanza delle seguenti regole:

1) Nessun elemento del network è analizzato fino a che gli stati di tutti i suoi input sono stati analizzati.
2) L' analisi dell'elemento di un network non è completa fino a che gli stati di tutti i suoi output non sono stati analizzati.
3) Come enunciato descrivendo l'editor FBD, un numero è assegnato automaticamente a tutti i network. All'interno di una POU, i network sono analizzati seguendo la sequenza del loro numero: il network `N` è analizzato prima del network `N+1`, a meno che non sia specificato diversamente mediante gli elementi di controllo dell'esecuzione.

11.3.3.2 COMBINAZIONE DI ELEMENTI

Gli elementi del linguaggio FBD devono essere interconnessi tramite linee di flusso di segnale.

Gli output dei blocchi non devono essere connessi insieme. In particolare, il costrutto "`Wired-OR`" del linguaggio LD non è permesso, mentre è richiesto un esplicito blocco Booleano "`OR`".

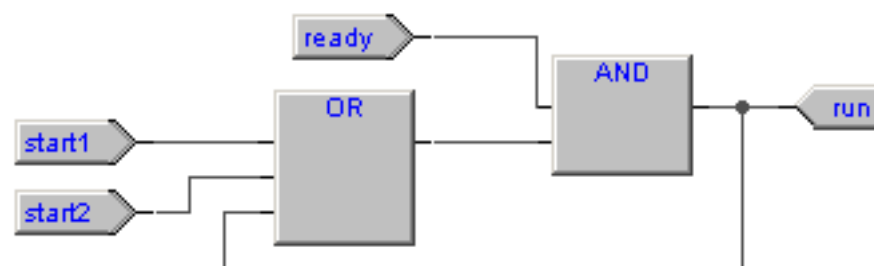
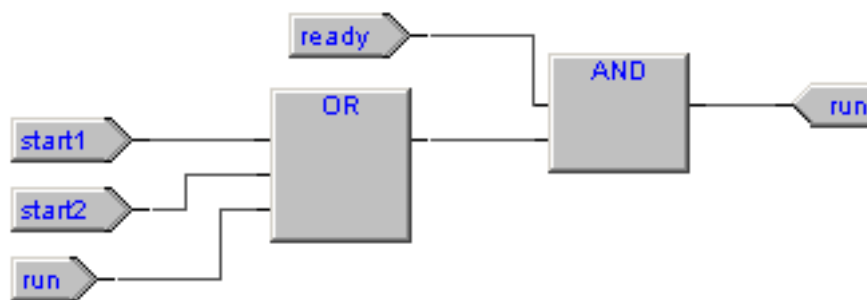
Feedback

Un percorso feedback esiste in un network quando l'output della funzione o del blocco funzione è utilizzato come input della funzione o del blocco funzione che lo precede nel network; la variabile associata è chiamata variabile feedback.

I percorsi feedback possono essere oggetto delle seguenti regole:

1) le variabili feedback devono essere inizializzate, e il valore iniziale è utilizzato durante la prima analisi del network. Vedere l'editor variabili globali, l'editor variabili locali o l'editor parametri per sapere come inizializzare le rispettive voci.
2) Una volta che l'elemento con una variabile di feedback come output è stato analizzato, il nuovo valore della variabile feedback è utilizzato fino all' analisi seguente dell'elemento.

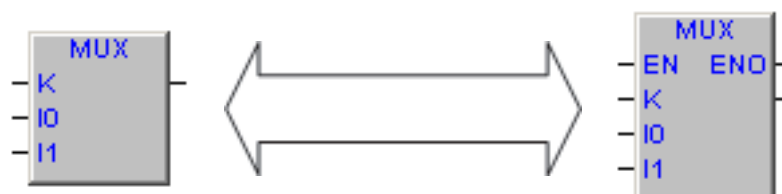
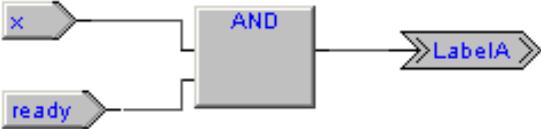
In tal caso, la variabile Booleana `RUN` è la variabile feedback nell'esempio mostrato di sotto.

Loop espliciti

Loop impliciti

11.3.4 ELEMENTI DI CONTROLLO DELL'ESECUZIONE

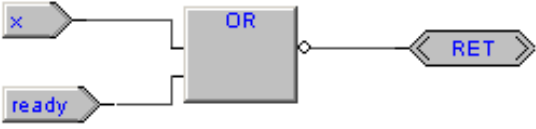
11.3.4.1 SEGNALI EN/ENO

Gli input addizionali Booleani EN (Enable) e ENO (Enable Out) caratterizzano i blocchi LogicLab, secondo le dichiarazioni

EN	ENO
VAR_INPUT EN: BOOL := 1; END_VAR	VAR_OUTPUT ENO: BOOL; END_VAR

Vedere la sezione delle proprietà di modifica dei blocchi per conoscere come aggiungere questi pins ad un blocco.

Quando si utilizzano queste variabili, l'esecuzione delle operazioni definite dal blocco sono controllate dalle regole seguenti:

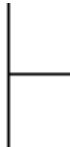
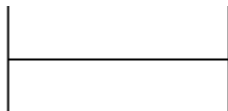
3) se il valore di EN è FALSE quando il blocco è richiamato, le operazioni definite dal corpo della funzione non sono eseguite e il valore di ENO è reimpostato a FALSE dal sistema del controller programmabile.
4) Altrimenti, il valore di ENO è impostato a TRUE dal sistema del controller programmabile, e le operazioni definite del corpo del blocco sono eseguite.

11.3.4.2 JUMP

I jump sono rappresentati da una linea di segnale Booleano che termina in una doppia freccia. La linea di segnale per una condizione di jump ha origine da una variabile Booleana, o da un output Booleano di una funzione o di un blocco funzione. Il trasferimento di un controllo del programma all'etichetta designata del network si verifica quando il valore Booleano della linea di segnale è TRUE; quindi, il salto incondizionale è una speciale variante del salto condizionale. Il target di un jump è un'etichetta del network all'interno della POU all'interno della quale si verifica un jump.

Simbolo / Esempio	Spiegazione
TRUE LabelA	Jump incondizionale
x LabelA	Jump condizionale
x ready AND LabelA	Esempio: network jump condizionale

11.3.4.3 RETURN CONDIZIONALI

- I risultati condizionali di funzioni e blocchi funzione sono implementati usando una costruzione RETURN come mostrato nella tabella sotto. L'esecuzione del programma è ritrasferita all'entità richiamante se l'input Booleano è TRUE, e continua nel modo normale se l'input Booleano è FALSE.
- I risultati incondizionali sono forniti dalla fine fisica della funzione o del blocco funzione.

Simbolo / Esempio	Spiegazione
x RET	Risultato condizionale
x ready OR RET	Example: network risultato condizionale

11.4 LADDER DIAGRAM (LD)

Questa sezione definisce la semantica del linguaggio LD.

11.4.1 POWER RAILS

Il network LD è delimitato nella parte sinistra da una linea verticale detta power rail sinistro, e nella parte destra da una linea verticale detta power rail destro. Il power rail destro può essere esplicito nell'implementazione LogicLab e sempre mostrato.

I due power rail sono sempre connessi con una linea orizzontale detta signal link. Tutti gli elementi LD dovrebbero essere posti e connessi al signal link.

Descrizione	Simbolo
Power rail sinistro (con link orizzontale attaccato)	
Power rail destro (con link orizzontale attaccato)	
Power rail connessi dal signal link	

11.4.2 ELEMENTI LINK E STATI

Gli elementi di link possono essere orizzontali o verticali. Lo stato degli elementi di link sarà su "ON" o "OFF", a seconda dei valori letterali Booleani 1 o 0, rispettivamente. Il termine link state è sinonimo di power flow.

Le seguenti proprietà si applicano agli elementi di link:

- lo stato del rail sinistro deve essere sempre considerato ON. Nessuno stato è definito per il rail destro.
- Un elemento di link orizzontale è indicato da una linea orizzontale. Un elemento di link orizzontale trasmette lo stato dell'elemento immediatamente alla sua sinistra all'elemento immediatamente alla sua destra.
- L'elemento di link verticale consiste in una linea verticale che si interseca con uno o più elementi di link orizzontale in ogni parte. Lo stato del link verticale rappresenta l'inclusivo OR degli stati ON dei link orizzontali nella sua parte sinistra, ovvero, lo stato del link verticale è: OFF se gli stati di tutti i link orizzontali attaccati alla sua sinistra sono:

  OFF se gli stati di tutti i link orizzontali attaccati alla sua sinistra sono OFF;

  ON se lo stato di uno o più dei link orizzontali attaccati alla sua sinistra è ON.
- Lo stato del link verticale è copiato su tutti i link verticali attaccati alla sua destra.
- lo stato del link verticale non è copiato su nessuno dei link orizzontali attaccati alla sua sinistra.

Descrizione	Simbolo
Link verticale con link orizzontali attaccati	

11.4.3 CONTATTI

Un contatto è un elemento che trasmette uno stato al link orizzontale nella sua parte destra che è uguale al Booleano AND dello stato del link orizzontale nella sua parte sinistra con una funzione appropriata di un input, output o variabili di memoria Booleani associati.

Un contatto non modifica il valore della variabile Booleana associata. I simboli di contatto standard sono mostrati nella tabella seguente.

Nome	Descrizione	Simbolo
Contatto normalmente aperto	Lo stato del link sinistro è copiato sul link destro se lo stato della variabile Booleana associata è ON. Altrimenti, lo stato del link destro è OFF.	| |
Contatto normalmente chiuso	Lo stato del link sinistro è copiato al link destro se lo stato della variabile Booleana associata è OFF. Altrimenti, lo stato del link destro è OFF.	|/|
Contatto sensibile al fronte di salita	Lo stato del link destro è ON dal momento della valutazione di questo elemento alla successiva valutazione quando viene rilevata una transizione da OFF a ON della variabile associata e lo stato del link sinistro è ON. Altrimenti, lo stato del link destro è OFF.	|P|
Contatto sensibile al fronte di discesa	Lo stato del link destro è ON dal momento della valutazione di questo elemento alla successiva valutazione quando viene rilevata una transizione da ON a OFF della variabile associata e lo stato del link sinistro è ON. Altrimenti, lo stato del link destro è OFF.	|N|

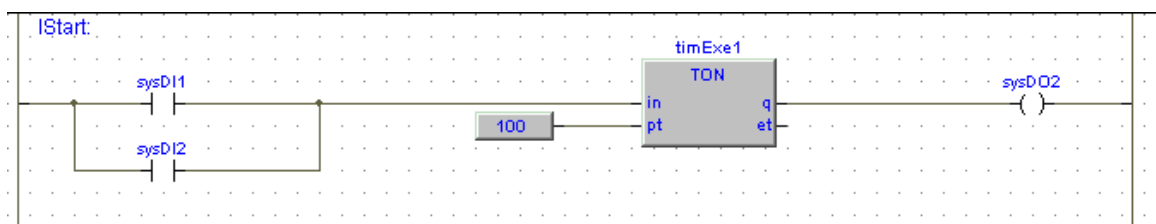
11.4.4 USCITE

Un'uscita copia lo stato del link alla sua sinistra al link alla sua destra senza alcuna modifica, e memorizza una funzione appropriata dello stato o della transizione del link sinistro nella variabile Booleana associata.

I simboli delle uscite standard sono mostrati nella tabella seguente.

Nome	Descrizione	Simbolo
Uscita	Lo stato del link sinistro è copiato nella variabile Booleana associata e nel link destro.	( )
Uscita negata	Lo stato del link sinistro è copiato sul link destro. L'inverso dello stato del link sinistro è copiato nella variabile Booleana associata, ovvero, se lo stato del link sinistro è OFF, allora lo stato della variabile associata è ON, e viceversa.	(/)
SET (latch) uscita	La variabile Booleana associata è reimpostata sullo stato OFF quando il link sinistro è sullo stato ON, e rimane reimpostato finchè non viene impostato da un'uscita SET.	(S)
RESET (unlatch) uscita	La variabile Booleana associata è reimpostata sullo stato OFF quando il link sinistro è sullo stato ON, e rimane reimpostato finchè non viene impostato da un'uscita SET.	(R)
Uscita sensibile al fronte di salita	Lo stato della variabile booleana associata è ON dal momento della valutazione di questo elemento alla successiva valutazione quando viene rilevata una transizione da OFF a ON del link sinistro.	(P)
Uscita sensibile al fronte di discesa	Lo stato della variabile booleana associata è ON dal momento della valutazione di questo elemento alla successiva valutazione quando viene rilevata una transizione da ON a OFF del link sinistro.	(N)

11.4.5 OPERATORI, FUNZIONI E BLOCCHI FUNZIONI

La rappresentazione di funzioni e blocchi funzione in linguaggio LD è simile a quella utilizzata per FBD. Verranno visualizzati su ogni blocco almeno un input Booleano e un output Booleano per calcolare il power flow attraverso il blocco come mostrato nella figura seguente.

11.5 STRUCTURED TEXT (ST)

Questa sezione definisce la semantica del linguaggio ST (Structured Text).

11.5.1 ESPRESSIONI

Un’espressione è un costrutto che, una volta calcolata, produce un valore corrispondente ad uno dei tipi di dati elencati nella tabella di tipi di dati elementari. LogicLab non impone nessuna restrizione sulla lunghezza massima delle espressioni.

Le espressioni sono composte da operatori e operandi.

11.5.1.1 OPERANDI

Un operando può essere un letterale, una variabile, un richiamo di funzione o un’altra espressione.

11.5.1.2 OPERATORI

Aprire la tabella degli operatori per visualizzare l’elenco di tutti gli operatori supportati da ST. L’analisi dell’espressione consiste nell’applicare gli operatori agli operandi in una sequenza definita dalle regole di precedenza degli operatori.

11.5.1.3 REGOLE DI PRECEDENZA DEGLI OPERATORI

Gli operatori hanno livelli diversi di precedenza, come specificato nella tabella degli operatori. L’operatore con la precedenza maggiore è applicato per primo in un’espressione, seguito dall’operatore con il grado di precedenza successivo, finchè l’analisi non è completa. Gli operatori con uguale livello di precedenza sono applicati come scritto nell’espressione, da sinistra a destra.

Per esempio se A,B,C e D sono di tipo INT con i rispettivi valori 1, 2, 3 e 4, allora:

```
A+B-C*ABS(D)
```

produce -9, and:

```
(A+B-C)*ABS(D)
```

produce 0.

Quando l’espressione ha due operandi, l’operando più a sinistra è analizzato per primo. Per esempio, nell’espressione

```
SIN(A)*COS(B)
```

l’espressione `SIN(A)` è analizzata per prima, seguita da `COS(B)`, seguita dall’analisi del prodotto.

Le funzioni sono richiamate come elementi dell’espressione che consistono nel nome della funzione seguito da un elenco di argomenti tra parentesi, come definito nella sezione corrispondente.

11.5.1.4 OPERATORI DEL LINGUAGGIO ST

Operazione	Simbolo	Precedenza
Parentesizzazione	(<expression>)	PIU' ALTA
Analisi di funzioni	<fname>(<arglist>)	.
Complemento di negazione	- NOT	.
Elevamento a potenza	**	.
Modulo moltiplica dividi	* / MOD	.
Addizione sottrazione	+ -	.
Comparazione	<, >, <=, >=	.
Uguaglianza Disuguaglianza	= <>	.
AND Boolean	AND	.
OR Exclusive Boolean	XOR	.
OR Boolean	OR	PIU' BASSA

11.5.2 DICHIARAZIONI IN ST

Tutte le istruzioni seguono le seguenti regole:

- terminano con il punto e virgola;
- a differenza di IL, il ritorno a capo o una nuova linea di carattere sono trattati allo stesso modo di un carattere di spazio;
- LogicLab non impone nessuna restrizione alla lunghezza massima delle istruzioni.

Le istruzioni ST, possono essere divise in classi, in rapporto alla loro semantica.

11.5.2.1 ASSEGNAMENTI

Semantica

Le istruzioni di assegnamento rimpiazzano il valore attuale di una variabile a elemento singolo/multiplo con il risultato del calcolo di un'espressione.

L'istruzione di assegnamento è anche utilizzata per assegnare il valore che si vuole che una funzione produca, ponendo il nome della funzione alla sinistra dell'operatore di assegnamento nel corpo di dichiarazione della funzione. Il valore prodotto dalla funzione è il risultato del calcolo più recente di questo assegnamento.

Sintassi

Un'istruzione di assegnamento consiste in un riferimento della variabile nella parte sinistra, seguita dall'operatore di assegnamento ":=", seguita dall'espressione da calcolare. In questo caso, l'istruzione

```
A := B ;
```

sarebbe usata per rimpiazzare il valore del dato singolo della variabile A dal valore attuale della variabile B se fossero entrambi di tipo INT.

Esempi

```
a := b ;
```

assegnamento

```
pCV := pCV + 1 ;
```

assegnamento

```
c := SIN( x );
```

assegnamento con richiamo di funzione

```
FUNCTION SIMPLE_FUN : REAL
variables declaration
...
function body
...
SIMPLE_FUN := a * b - c ;
END_FUNCTION
```

assegnare il valore dell'output ad una funzione.

11.5.2.2 DICHIARAZIONE DI FUNZIONI E BLOCCHI FUNZIONE

Semantica

- Le funzioni sono richiamate come elementi dell'espressione che consistono nel nome della funzione seguito da un elenco di argomenti tra parentesi. Ciascun argomento può essere letterale, una variabile o un'espressione complessa arbitraria.
- I blocchi funzione sono richiamati da un'istruzione che consiste nel nome della ricorrenza del blocco funzione seguito da un elenco degli argomenti tra parentesi..E' supportato sia il richiamo con la lista formale degli argomenti sia l'assegnamento di argomenti.
- RETURN: le istruzioni di controllo di funzione e di blocco funzione consistono nei meccanismi per richiamare i blocchi funzione e per riportare il controllo all'entità richiamata prima della fine fisica della funzione o del blocco funzione. L'istruzione RETURN causa l'uscita anticipata dalla funzione o dal blocco funzione (per esempio, come risultato dell'analisi di un'istruzione IF).

Sintassi

1) Funzione:

```
dst_var := function_name( arg1, arg2 , ... , argN );
```

2) Blocco funzione con elenco argomenti formali:

```
instance_name(    var_in1 := arg1 ,
                  var_in2 := arg2 ,
                  ... ,
                  var_inN := argN );
```

3) Blocco funzione con assegnamento di argomenti:

```
instance_name.var_in1 := arg1;
...
instance_name.var_inN := argN;
instance_name();
```

4) Istruzione di controllo di funzione e blocco funzione:

```
RETURN;
```

Esempi

```
CMD_TMR( IN := %IX5,
 PT:= 300 ) ;
```

richiamo FB con elenco di argomenti formali:

```
IN := %IX5 ;
PT:= 300 ;
CMD_TMR() ;
```

richiamo FB con assegnamento di argomenti:

```
a := CMD_TMR.Q;
```

utilizzo di output FB:

```
RETURN ;
```

uscita anticipata dalla funzione o dal blocco funzione.

11.5.2.3 ISTRUZIONI DI SELEZIONE

Semantica

Le istruzioni di selezione includono le istruzioni IF e CASE. Un'istruzione di selezione seleziona uno (o un gruppo) di sue istruzioni componenti di esecuzione basate su una condizione specifica.

- IF: l'istruzione IF specifica che un gruppo di istruzioni deve essere eseguito solo se l'espressione Booleana associata restituisce il valore TRUE. Se la condizione è falsa, nessuna istruzione sarà eseguita, o verrà eseguito il gruppo di istruzioni che segue la keyword ELSE (o la keyword ELSIF se la sua condizione Booleana associata è TRUE).
- CASE: l'istruzione CASE consiste in un' espressione che valuta il valore della variabile di tipo DINT (il "selettore") e una lista di gruppi di istruzioni, ogni gruppo viene etichettato da uno o più interi o gamma di valori interi. Viene eseguito il primo gruppo di istruzioni che contiene il valore del selettore. Se il valore del selettore non è presente in nessun caso, viene eseguita la sequenza di istruzioni che segue la keyword ELSE (se è presente nell'istruzione CASE). Altrimenti, non verrà eseguita nessuna delle sequenze di istruzioni.

LogicLab non impone nessuna restrizione al numero massimo di selezioni possibili nell'istruzione CASE.

Sintassi

Notare che le parentesi quadrate includono il codice opzionale, mentre le graffe comprendono segmenti di codice ripetibili.

1) IF:

```
IF expression1 THEN
stat_list
[ { ELSIF expression2 THEN
stat_list } ]
ELSE
stat_list
END_IF ;
```

2) CASE:

```
CASE expression1 OF
intv [ {, intv } ] :
stat_list
{ intv [ {, intv } ] :
```

```
stat_list }
[ ELSE
stat_list ]
END_CASE ;
intv being either a constant or an interval: a or a..b
```

Esempi

Istruzione IF:

```
IF d 0.0 THEN
nRoots := 0 ;
ELSIF d = 0.0 THEN
nRoots := 1 ;
x1 := -b / (2.0 * a) ;
ELSE
nRoots := 2 ;
x1 := (-b + SQRT(d)) / (2.0 * a) ;
x2 := (-b - SQRT(d)) / (2.0 * a) ;
END_IF ;
```

Istruzione CASE:

```
CASE tw OF
1, 5:
display := oven_temp ;
2:
display := motor_speed ;
3:
display := gross_tare;
4, 6..10:
display := status(tw - 4) ;
ELSE
 display := 0;
 tw_error := 1;
END_CASE ;
```

11.5.2.4 ISTRUZIONI DI ITERAZIONE

Le istruzioni di iterazione specificano che il gruppo di istruzioni associate sono eseguite ripetutamente. L'istruzione FOR è utilizzata se il numero di iterazioni può essere determinato in precedenza; altrimenti, sono utilizzati i costrutti WHILE e REPEAT.

- FOR: l'istruzione FOR indica che la sequenza di istruzioni è eseguita ripetutamente, fino alla keyword END_FOR, mentre una progressione di valori è assegnata alla variabile di controllo del ciclo FOR. Il valore iniziale e il valore finale della variabile di controllo, sono espressioni dello stesso tipo di intero (per esempio, SINT, INT o DINT) e non possono essere alterati da nessuna delle istruzioni di ripetizione. L'istruzione FOR incrementa o decrementa il valore della variabile di controllo dal valore iniziale al valore finale con incrementi determinati dal valore di una espressione; questo valore di default è 1. Il test per la condizione di termine è fatto all'inizio di ogni iterazione, così che la sequenza di istruzione non è eseguita se il valore iniziale eccede quello finale.

- WHILE: l'istruzione WHILE fa sì che la sequenza di istruzioni sia eseguita ripetutamente fino alla keyword END_WHILE e finchè l'espressione Booleana associata risulta false. Se l'espressione è inizialmente falsa, allora il gruppo di istruzioni non viene eseguito.
- REPEAT: l'istruzione REPEAT fa sì che la sequenza di istruzioni sia eseguita ripetutamente (almeno una volta) fino alla keyword UNTIL e finchè la condizione Booleana associata è vera.
- EXIT: l'istruzione EXIT è usata per terminare le iterazioni prima che la condizione di termine sia soddisfatta. Quando l'istruzione EXIT è posizionata all'interno dei costrutti iterativi annidati, l'uscita avviene dal ciclo più interno in cui si trova l'EXIT, cioè, il controllo passa all'istruzione successiva dopo il primo terminatore del ciclo (END_FOR, END_WHILE o END_REPEAT) che segue l'istruzione EXIT.

Nota: le istruzioni WHILE e REPEAT non possono essere utilizzate per raggiungere la sincronizzazione interprocessuale, per esempio come un "wait loop" con una condizione di termine determinata esternamente. Per questo scopo devono essere utilizzati gli elementi SFC definiti.

Sintassi

Notare che le parentesi quadre includono codice opzionale, mentre quelle graffe includono segmenti di codice ripetibile.

1) FOR:

```
FOR control_var := init_val TO end_val [ BY increm_val ] DO
stat_list
END_FOR ;
```

2) WHILE:

```
WHILE expression DO
stat_list
END_WHILE ;
```

3) REPEAT:

```
REPEAT
stat_list
UNTIL expression
END_REPEAT ;
```

Esempi

FOR statement:

```
j := 101 ;
FOR i := 1 TO 100 BY 2 DO
 IF arrvals[i] = 57 THEN
j := i ;
 EXIT ;
 END_IF ;
END_FOR ;
```

WHILE statement:

```
j := 1 ;
WHILE j <=100 AND arrvals[i] <>  57 DO
j := j + 2 ;
END_WHILE ;
```

REPEAT statement:

```
j := -1 ;
```

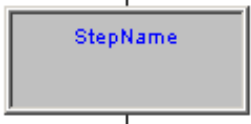
```
REPEAT
 j := j + 2 ;
UNTIL j = 101 AND arrvals[i] = 57
END_REPEAT ;
```

11.6 SEQUENTIAL FUNCTION CHART (SFC)

Questa sezione definisce gli elementi SFC per strutturare l'organizzazione interna di una POU PLC, scritta in uno dei linguaggi definiti dallo standard, con lo scopo di eseguire funzioni di controllo sequenziali. Le definizioni in questa sezione derivano da IEC848, con i cambiamenti necessari per convertire le rappresentazioni di uno standard di documentazione ad un set di elementi di controllo di esecuzione per una POU PLC.

Dal momento che gli elementi SFC richiedono la memorizzazione di informazione di stato, le uniche POU che possono essere strutturate con questi elementi sono i blocchi funzione e i programmi.

Se una qualsiasi parte di una POU è suddivisa in elementi SFC, l'intera POU sarà così suddivisa. Se non è fornita nessuna suddivisione SFC per una POU, l'intera POU è considerata come un'azione singola che si esegue sotto il controllo dell'entità che l'ha richiamata.

Elementi SFC

Gli elementi SFC forniscono mezzi di suddivisione di una POU PLC in una serie di stati e transizioni interconnesse tramite link diretti. Un set di azioni è associato ad ogni stato, e ad ogni transizione è associata una condizione di transizione.

11.6.1 STEP

11.6.1.1 DEFINIZIONE

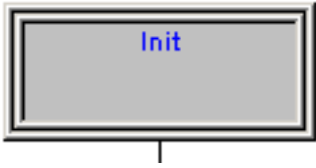
Uno stato rappresenta una situazione in cui il comportamento della POU, con rispetto dei suoi input e output, segue una serie di regole definite dalle azioni associate dello stato. Uno stato è attivo o inattivo. In ogni momento dato, lo stato della POU è definito da una serie di stati attivi e dal valore delle sue variabili interne e output.

Uno stato è rappresentato graficamente da un blocco che contiene il nome dello stato nella forma di un identificatore. Il link(s) diretto nello stato può essere rappresentato graficamente da una linea verticale attaccata alla parte superiore dello stato. Il link(s) diretto fuori dallo stato può essere rappresentato da una linea verticale attaccata alla parte inferiore dello stato.

Rappresentazione	Descrizione
StepName	Step (rappresentazione grafica con link diretti)

LogicLab non impone nessuna restrizione al numero massimo di stati per SFC.

Step flag

La step flag (stato inattivo o attivo di uno stato) può essere rappresentata dal valore logico della variabile Booleana ***_x, dove *** è il nome dello stato. Questa variabile Booleana ha valore TRUE quando lo stato corrispondente ha è attivo, e FALSE quando è inattivo. La visibilità dei nomi degli stati e delle step flag è locale alla POU in cui compaiono gli stati.

Rappresentazione	Descrizione
`Step Name_x`	Step flag = TRUE when `Step Name_x` is active = FALSE otherwise

Gli utenti non possono assegnare un valore direttamente allo stato di uno step.

11.6.1.2 STEP INIZIALI

Lo stato iniziale della POU è rappresentato dai valori iniziali delle sue variabili interne e output, e dalla sua serie di stati iniziali, ovvero, gli stati che sono inizialmente attivi. Ogni network SFC, o il suo equivalente testuale, ha esattamente uno stato iniziale. Uno stato iniziale può essere disegnato graficamente col bordo a doppia linea, come mostrato sotto. Per l'inizializzazione del sistema, lo stato iniziale di default è FALSE per gli stati ordinari e TRUE per quelli iniziali.

LogicLab non può compilare un network SFC che non contiene esattamente uno stato iniziale.

Rappresentazione	Descrizione
Init	Initial step (rappresentazione grafica con link diretti)

11.6.1.3 AZIONI

Un'azione può essere:

- una collezione di istruzioni in linguaggio IL;
- una collezione di network in linguaggio FBD;
- una collezione di scalini in linguaggio LD;
- una collezione di istruzioni in linguaggio ST;
- una SFC organizzata come specificato in questa sezione.

Ad ogni stato possono essere associate nessuna o più azioni. Le azioni sono dichiarate tramite gli elementi di struttura testuale elencati qui sotto.

Elementi strutturali	Descrizione

`STEP StepName :`
`(* Step body *)`
`END_STEP`
	Step (forma testuale)

`INITIAL_STEP StepName :`
`(* Step body *)`
`END_STEP`
	Initial step (forma testuale)

Un elemento di struttura di questo genere esiste nel file Isc di ogni stato che ha almeno un'azione associata.

11.6.1.4 QUALIFICATORI ACTION

Il tempo in cui è eseguita un'azione associata ad uno step dipende dal suo qualificatore di azioni.

LogicLab implementa i seguenti qualificatori di azioni.

Qualificatore	Descrizione	Significato
N	Non-stored (null qualifier).	L'azione è eseguita per tutto il tempo in cui lo step rimane attivo.
P	Pulse.	L'azione è eseguita solo la prima volta all'attivazione dello step, indipendentemente dal numero di cicli in cui lo step rimane attivo.

Se uno stato non ha azioni associate, allora si considera che abbia una funzione WAIT, ovvero, aspetta che la condizione di transizione del successore diventi TRUE.

11.6.1.5 JUMP

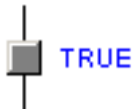
I link diretti si dirigono solo verso il basso. Per questo motivo, se si vuole ritornare ad uno stato superiore da uno inferiore, non è possibile disegnare un filo logico dall'ultimo al primo. Esiste un tipo di blocco speciale, chiamato Jump (salto) che permette di implementare questo genere di transizione.

Un blocco Jump è logicamente equivalente ad uno step, poiché devono essere sempre separati da una transizione. L'unico effetto di un Jump è l'attivazione della step flag dello step precedente e l'attivazione della flag dello step in cui porta.

Rappresentazione	Descrizione
JMP	Jump (link logico allo step di destinazione)

11.6.2 TRANSITION

11.6.2.1 DEFINIZIONI

Una transizione rappresenta la condizione con cui il controllo passa da uno o più step precedenti la transizione ad uno o più step successivi lungo il link diretto corrispondente.

La transizione è rappresentata come un quadratino grigio sul link diretto verticale.

La direzione di evoluzione che segue i link diretti è dalla parte inferiore dello step precedente verso la parte superiore dello step successivo.

11.6.2.2 CONDIZIONI DI TRANSIZIONE

Ogni transizione ha una condizione di transizione associata che è il risultato del calcolo di un'espressione Booleana singola. Una condizione di transizione sempre true è rappresentata dalla keyword TRUE, mentre una condizione di transizione sempre false è simbolizzata dalla keyword FALSE.

Una condizione di transizione può essere associata con una transizione attraverso uno dei seguenti mezzi:

Rappresentazione	Descrizione
TRUE	Ponendo la costante Booleana appropriata {TRUE, FALSE} adiacente al link diretto verticale.

Rappresentazione	Descrizione
VarName	Dichiarando una variabile Booleana, il cui valore determina se la transizione è verificata.
ProgName	Scrivendo un segmento di codice, in uno dei linguaggi supportati da LogicLab, tranne SFC. Il risultato dell'analisi di un codice di questo tipo determina la condizione di transizione.

La visibilità di un nome di transizione è locale alla POU in cui la transizione è posizionata.

11.6.3 REGOLE DI EVOLUZIONE

Introduzione

La situazione iniziale di un network SFC è caratterizzata dallo step iniziale che si trova nello stato attivo sull'inizializzazione del programma o del blocco funzione che contiene il network.

Le evoluzioni degli stati attivi di stati avvengono sui link diretti quando sono causate dalla verifica di una o più transizioni.

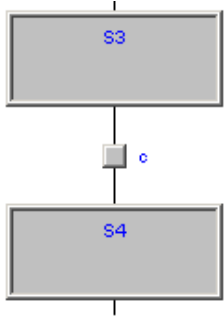
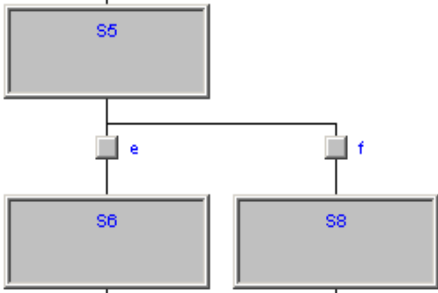
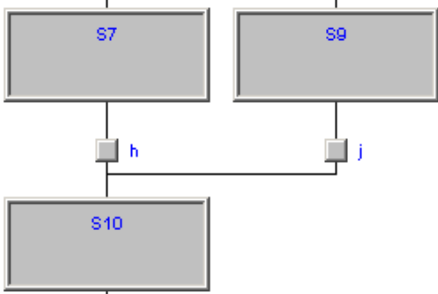
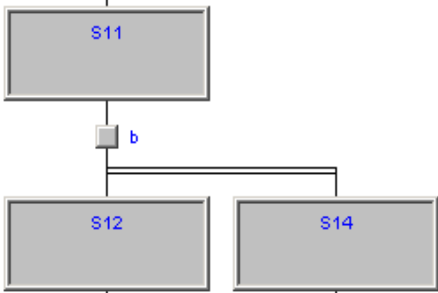
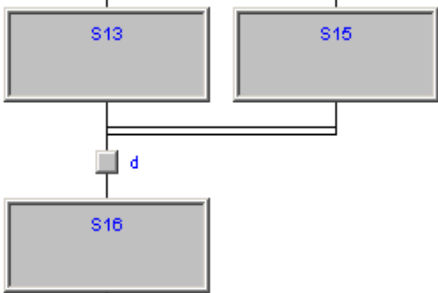
Una transizione è abilitata quando tutti gli step precedenti, connessi al simbolo di transizione corrispondente tramite link diretti, sono attivi. La verifica di una transizione avviene quando la transizione è abilitata e la condizione di transizione associata è true.

La verifica di una transizione causa la disattivazione (o "reimpostazione") di tutti gli step immediatamente precedenti connessi al simbolo di transizione corrispondente tramite link diretti, seguita dall'attivazione di tutti gli step immediatamente seguenti.

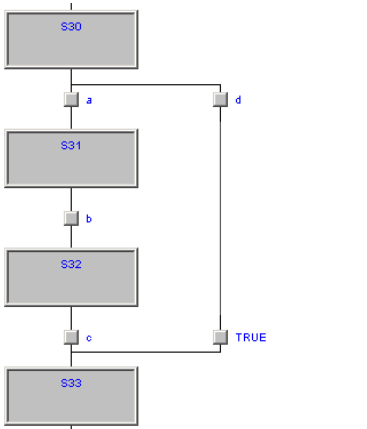
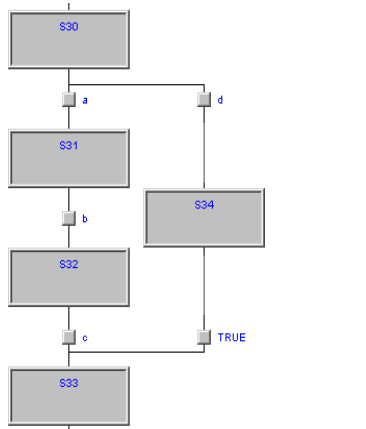
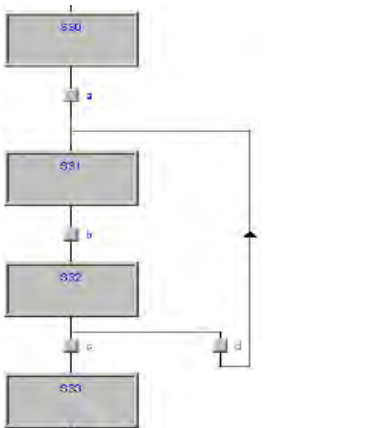
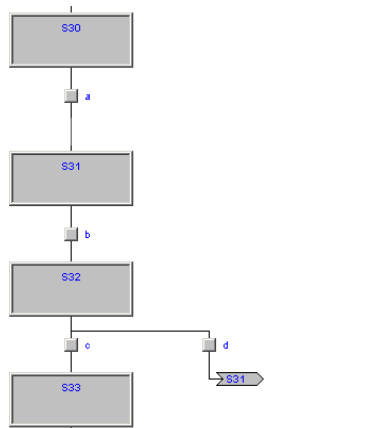
L'alternanza Stato/Transizione e Transizione/Stato è sempre mantenuta nelle connessioni di elementi SFC, ovvero:

- due stati non sono mai direttamente collegati; sono sempre separati da una transizione;
- due transizioni non sono mai collegate direttamente; sono sempre separate da uno step.

Quando la verifica di una transizione porta all'attivazione di numerosi step allo stesso tempo, le sequenze alle quali questi step appartengono sono chiamate sequenze simultanee. Dopo la loro attivazione simultanea, l'evoluzione di ognuna di queste sequenze diventa indipendente. Per enfatizzare la speciale natura di questi costrutti, la divergenza e la convergenza di sequenze simultanee è indicata da una linea orizzontale doppia.

Il tempo di verifica di una transizione può essere teoricamente considerato tanto corto quanto si vuole, ma non può mai essere zero.

In pratica, il tempo di verifica sarà imposto dall'implementazione PLC: numerose transizioni che possono essere verificate simultaneamente saranno verificate simultaneamente, entro le restrizioni di tempo della particolare implementazione PLC le restrizioni di priorità definite nella tabella di evoluzione della sequenza. Per la stessa ragione, la durata di attività di uno step non può mai considerarsi essere zero.

Il test della condizione di transizione del successore di uno step attivo non sarà eseguito finchè gli effetti dell'attivazione dello step non si saranno propagati attraverso tutta la POU in cui lo step è dichiarato.

Tabella di evoluzione della sequenza

Questa tabella definisce la sintassi e la semantica delle combinazioni di step e transizioni ammesse.

Esempio	Regola
S3 c S4	Transizione normale Un'evoluzione dallo stato S3 allo stato S4 avviene se e soltanto se lo stato S3 è attivo e la condizione di transizione è TRUE.
S5 e f S6 S8	Transizione divergente Avviene un'evoluzione da S5 a S6 se e soltanto se S5 è attivo e la condizione di transizione e è TRUE, o da S5 a S8 se e soltanto se S5 è attivo e f è TRUE e e è FALSE.
S7 S9 h j S10	Transizione convergente Avviene un'evoluzione da S7 a S10 soltanto se S7 è attivo e la condizione di transizione h è TRUE, o da S9 a S10 soltanto se S9 è attivo e j è TRUE.
S11 b S12 S14	Transizione divergente simultanea Avviene un'evoluzione da S11 a S12, S14,… soltanto se S11 è attivo e la condizione di transazione b associata alla transizione comune è TRUE. Dopo l'attivazione simultanea di S12, S14, ecc…, l'evoluzione di ciascuna sequenza procede indipendentemente.
S13 S15 d S16	Transizione convergente simultanea Avviene un'evoluzione da S13, S15,… a S16 soltanto se tutti gli stati che sono sopra e sono connessi alla linea orizzontale doppia sono attivi e la condizione di transizione d associata alla transizione comune è TRUE.

Esempi

Schema non valido	Schema equivalente permesso	Note
		Comportamento previsto: avviene un'evoluzione da S30 a S33 se a è FALSE e d è TRUE. Lo schema nella colonna più a sinistra non è valido perché le condizioni d e TRUE sono direttamente collegate.
		Comportamento previsto: avviene un'evoluzione da S32 a S31 se c è FALSE e d è TRUE. Lo schema nella colonna più a sinistra non è valido perché i link diretti si dirigono solo verso il basso. Le transizioni verso l'alto possono essere eseguite tramite i blocchi jump.

11.7 ESTENSIONI DI LINGUAGGIO IN LOGICLAB

LogicLab presenta alcune estensioni allo standard IEC 61131-3, per arricchire ulteriormente il linguaggio e adattarsi ai differenti stili di codice.

11.7.1 MACRO

LogicLab implementa le macro nello stesso modo del preprocessore di linguaggio C.

Le macro possono essere definite usando la seguente sintassi:

```
MACRO <macro name>
PAR_MACRO
      <parameter list>
END_PAR
<macro body>
END_MACRO
```

Notare che l'elenco dei parametri potrebbe essere eventualmente vuoto, facendo distinzione in questo modo oggetti tipo macro, che non prendono parametri, e funzioni tipo macro, che prendono parametri.

Un esempio concreto di una definizione macro è la seguente, che prende due Byte e compone una word 16-bit:

```
MACRO MAKEWORD
    PAR_MACRO
    lobyte;
    hibyte;
    END_PAR
    { CODE:ST }
    lobyte + SHL( TO_UINT( hibyte ), 8 )
END_MACRO
```

Ogni qualvolta il nome della macro appare nel codice sorgente, è rimpiazzato (insieme alla lista dei parametri correnti, in caso di macro tipo funzione) con il corpo della macro. Per esempio, data la definizione della macro MAKEWORD e il frammento di codice di Structured Text seguente:

```
w := MAKEWORD( b1, b2 );
```

il pre-processore macro lo espande a

```
w := b1 + SHL( TO_UINT( b2 ), 8 );
```

11.7.2 PUNTATORI

I puntatori sono un tipo speciale di variabili che si comportano come un riferimento ad un'altra variabile (la variabile 1puntata). Il valore di un puntatore è, infatti, l'indirizzo della variabile puntata; per accedere ai dati memorizzati nell'indirizzo puntato, i puntatori possono essere dereferenziati.

La dichiarazione di un puntatore richiede la stessa sintassi usata nella dichiarazione di variabili, dove il nome del tipo è il nome del tipo della variabile puntata preceduto dal segno @:

```
VAR
<pointer name> : @<pointed variable type name>;
END_VAR
```

Per esempio, la dichiarazione di un puntatore ad una variabile REAL è la seguente:

```
VAR
px : @REAL;
END_VAR
```

Un puntatore può essere assegnato ad un altro puntatore o ad un indirizzo. Uno speciale operatore, ADR, è disponibile per recuperare l'indirizzo di una variabile.

```
    px := py;         (* px and py are pointers to REAL (that is, vari-
ables of type @REAL) *)
    px := ADR( x )    (* x is a variable of type REAL *)
    px := ?x          (* ? is an alternative notation for ADR *)
```

L'operatore @ è utilizzato per togliere il riferimento al puntatore, quindi per accedere alla variabile puntata.

```
px := ADR( x );
@px := 3.141592;  (* the approximate value of pi is assigned to x *)
pn := ADR( n );
n := @pn + 1;           (* n is incremented by 1 *)
```

Prestare attenzione al fatto che l'uso improprio dei puntatori è potenzialmente pericoloso: infatti, i puntatori possono puntare a qualsiasi posizione arbitraria, che potrebbe causare effetti indesiderati.